Round Optimal Black-Box “Commit-and-Prove”

Dakshita Khurana[1], Rafail Ostrovsky[⋆2], and Akshayaram Srinivasan[⋆⋆3]

[1] Microsoft Research, New England
[2] UCLA
[3] UC Berkeley

Abstract. Motivated by theoretical and practical considerations, an important line of research is to design secure computation protocols that only make black-box use of cryptography. An important component in nearly all the black-box secure computation constructions is a *black-box commit-and-prove* protocol. A commit-and-prove protocol allows a prover to commit to a value and prove a statement about this value while guaranteeing that the committed value remains hidden. A black-box commit-and-prove protocol implements this functionality while only making black-box use of cryptography.

In this paper, we build several tools that enable constructions of *round-optimal*, black-box commit and prove protocols. In particular, assuming injective one-way functions, we design the first round-optimal, black-box commit-and-prove arguments of knowledge satisfying strong privacy against malicious verifiers, namely:

- Zero-knowledge in four rounds and,
- Witness indistinguishability in three rounds.

Prior to our work, the best known black-box protocols achieving commit-and-prove required more rounds.

We additionally ensure that our protocols can be used, if needed, in the *delayed-input* setting, where the statement to be proven is decided only towards the end of the interaction. We also observe simple applications of our protocols towards achieving black-box four-round constructions of extractable and equivocal commitments.

We believe that our protocols will provide a useful tool enabling several new constructions and easy round-efficient conversions from non-black-box to black-box protocols in the future.

[⋆] Research supported in part by NSF grant 1619348, DARPA SafeWare subcontract to Galois Inc., DARPA SPAWAR contract N66001-15-1C-4065, US-Israel BSF grant 2012366, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. The views expressed are those of the authors and do not reflect position of the Department of Defense or the U.S. Government.

[⋆⋆] Research supported in part from Sanjam Garg's 2017 AFOSR YIP Award, DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the author and do not reflect the official policy or position of the funding agencies.

1 Introduction

Secure computation [42, 13] allows a set of mutually distrusting parties to compute a joint function of their private inputs such that nothing else apart from the function's output is leaked. The constructions of secure computation protocols (where the majority of the parties can be corrupted) may make use of cryptographic primitives in one out of the following two ways. The construction can either make *black-box* use of the primitive by referring only to the input/output behavior of that primitive or it can make *non-black-box* use of the primitive by using the code computing this primitive.

Typically, non black-box use of a cryptographic primitive is made to protect against malicious adversaries who may deviate arbitrarily from the protocol specification. In such scenarios, a zero-knowledge proof [14] showing correct computation of this primitive (which in turn requires access to the code computing this primitive) is used. This part is computationally expensive and further, the complexity of this step depends on the actual implementation of this cryptographic functionality.

The advantage of black-box constructions is that their complexity is independent of the complexity of implementation of the underlying primitive. In fact, such protocols are sometimes considered as the first step towards practical implementations. There has been an impressive body of research [25, 19, 7, 27, 28, 36, 15, 26, 16] on constructing secure computation protocols that make black-box use of underlying primitives. However, most of these works incur several additional rounds of interaction when compared to non-black-box protocols.

A very natural question, which is still far from being resolved, is whether there exist *black-box* protocols that match the exact round complexity of their non-black-box counterparts.[4] In this paper, we ask this question for a key cryptographic functionality that lies at the center of nearly all black-box constructions: *the commit-and-prove functionality.*

1.1 Commit-and-Prove Functionalities

A "commit-and-prove" functionality [13, 6] is generally used to prevent malicious behaviour by forcing participants to prove correctness of their protocol messages w.r.t. the committed inputs. Informally, a commit-and-prove functionality allows a party to commit to some secret value x and prove that value satisfies some predicate P. In order to maintain secrecy, such a proof must additionally hide the secret input x, in other words it must be *zero-knowledge.*

Very roughly, any commit-and-prove protocol is said to be zero-knowledge if there exists an associated simulator that given the commitment externally, is able to generate a proof without access to the witness (or the value being committed

[4] Two notable exceptions are the works of [34] and [18] who construct round-optimal secure computation, and non-malleable commitments respectively via black-box use of cryptography. However, these works developed techniques very specific to their respective settings.

via the commitment). We note that such protocols have been a core primitive in nearly all previous works on obtaining black-box constructions, including [25, 27, 36, 16]. In addition to zero-knowledge, we also consider the weaker privacy property of witness indistinguishability [9].[5]

Despite the above mentioned fascinating advances in constructing such protocols, we still do not know round-optimal black-box constructions of zero-knowledge commit-and-prove functionalities. Indeed, in the black-box regime, the best known result is due to Hazay and Venkitasubramaniam [21] which requires 6 rounds of interaction. In fact, when not restricted to black-box use of primitives, we have known for more than 25 years that four rounds are necessary [12][6] and sufficient [5, 8] for constructing zero-knowledge commit-and-prove arguments.

However, so many years later, in the regime of *black-box commit-and-prove*, the following question is still open:

"Do there exist round-optimal, black-box commit-and-prove zero-knowledge protocols?"

1.2 Our Results

We provide a positive answer to this question. In particular, assuming injective one-way functions, we construct the first:

- Four round black-box commit-and-prove zero-knowledge arguments of knowledge against malicious verifiers, and
- Three round black-box commit-and-prove witness-indistinguishable arguments of knowledge against malicious verifiers. These commitments satisfy only a weaker notion of binding (that we call 1-of-2 binding), which nevertheless suffices for all our applications, and which we detail in Definition 4.

Our protocols satisfy correctness and soundness even in the *delayed-input* setting, where the predicate to be proved can be decided even in the last round of the protocol, however, the witness or message to be committed must be known before the prover sends his first message. Additionally, as simple applications of these protocols, we give the first constructions of four round extractable and equivocal commitments that only make black-box use of injective one-way functions.

Discussion. Our construction makes non-black-box use of the predicate, similar to all previous constructions in this line of work. With respect to constructions making black-box use of the predicate, we would like to point to the negative result of Rosulek [39] which shows that any (honest-verifier) zero-knowledge

[5] Please refer to Section 3 for a formal definition of witness indistinguishable commit-and-prove protocols.

[6] Due to limitations on the round-complexity required to implement existing non black-box techniques, we restrict ourselves to black-box reductions in this paper.

argument for the NP language $L = \{x : \exists w s.t. f(w) = x\}$ and f is a one-way function must make use of the code of the function f.

We note that similar to previous work using MPC-in-the-head [IKOS07] and follow-ups, our techniques can be used directly to build black-box protocols in cases where the predicate is information theoretic. There are several settings in literature where the predicate is indeed information theoretic. A few simple examples include:

- A commit-and-prove protocol for checking equality of two committed values (This is in fact used in our construction of equivocal commitments).
- A commit-and-prove protocol for checking that one committed value corresponds to a fixed polynomial evaluated on a different committed value (Eg, in the ZK arguments that achieve four round non-malleable commitments in GRRV14.)
- Comparison queries or range proofs, showing that a committed value lies in a certain fixed range (Such proofs have become increasingly popular in recent years).
- These techniques may also be relevant to distributed secure protocols with information-theoretic guarantees.

Furthermore, as we note in the paper, in many other situations, where the predicate itself involves cryptography, cut-and-choose techniques have been extensively explored (Please see [31, 41, 33, 40, 32, 22, 23, 30, 1] and references therein). Specifically, the works of [16, 17, 35] used cut-and-choose to separate such predicates into cryptographic components, for which malicious security was obtained using cut-and-choose, and information-theoretic components, for which tailored commit-and-prove protocols were built. In this paper, we concentrate on the latter and build round-optimal, black-box commit-and-prove ZK protocols to generically solve the problem of commit-and-prove for information-theoretic predicates. As simple applications of these results, in the paper, we construct the first four round extractable and equivocal commitments from injective one-way functions.

1.3 Related Works

Goldreich and Krawcyzk [12] showed that four rounds are necessary to construct zero-knowledge argument system that make black-box use of a verifier for languages outside of BPP. Bellare et al. [5] and Feige et al. [8] gave protocols that matches this round complexity from the minimum assumption that one-way functions exist.

The commit and prove functionality was first used implicitly in [13] and was later formalized in [6]. The constructions given in these works made non-black-box use of one-way functions. A constant round black-box commit and prove zero-knowledge proof was implicit in the work of Ishai et al. [27] assuming collision resistant hash functions. Later works of [16, 17, 35] improved the concrete round complexity of this construction and also constructed zero-knowledge argument systems from one-way functions. More recently, Hazay and Venkitasubramaniam [21] constructed a six-round black-box commit and prove zero-knowledge argument from injective one-way functions. This work represents the state of the art in terms of round complexity of black-box commit and prove.

2 Our Techniques

Our starting point is the work of Hazay and Venkitasubramaniam [21], who constructed three-message black-box commit-and-prove ZK protocols with constant soundness, by making use of *robust offline/online randomized encodings.*

Starting point: Robust Offline/Online Encoding. A randomized encoding [42, 24, 2, 3, 4] of a boolean circuit f is a function $\widehat{f}$ along with a decoding algorithm Dec such that for any input x in the domain of the function f, with overwhelming probability it holds that $\mathrm{Dec}(\widehat{f}(x; U_m)) = f(x)$, where U_m denotes the uniform distribution over m bits. Moreover, the encoding $\widehat{f}$ required to satisfy computational privacy, meaning that the encoding $\widehat{f}(x, U_m)$ reveals no information about x and f, except $f(x)$. A randomized encoding is called offline/online if it has two components: an offline component that does not depend on the input, and an online component which is a function of the input. We will denote these by two functions $\widehat{f}_{\mathrm{off}}$ and $\widehat{f}_{\mathrm{on}}$ such that $\widehat{f}(x; r) = (\widehat{f}_{\mathrm{off}}(r), \widehat{f}_{\mathrm{on}}(x, r))$. Such an encoding is called *robust*, if additionally the following is true: when there exists no x such that $f(x) = a$, then for any r, there does not exist any z such that $\mathrm{Dec}(\widehat{f}_{\mathrm{off}}(r), z) = a$. The work of [21] showed that robust randomized encodings can be instantiated in multiple ways, including the use of adaptive garbled circuits.

Black-Box Commit and Prove with Constant Soundness. Let us now explain how the work of [21] used offline/online randomized encodings to construct black-box commit and prove ZK proofs with constant soundness error. Additionally, if the encodings are robust then this zero-knowledge proof satisfies correctness and soundness in the *delayed-input* setting.

The prover $\mathcal{P}$ has a message m and wants to convince the verifier that $\phi(m) = 1$ where ϕ is some predicate. The protocol is as follows:

1. In the first round, $\mathcal{P}$ secret shares m into two shares m_0 and m_1. It then constructs a function f which has hardwired a secret share m_0 of m, and obtains as input the other share m_1 and the predicate ϕ. This function outputs $(1, \phi, m_1)$ if and only if $\phi(m_0 \oplus m_1) = 1$; otherwise it outputs $\perp$. It constructs an offline encoding of this function $\widehat{f}_{\mathrm{off}}(r)$ and sends $\widehat{f}_{\mathrm{off}}(r)$ and also sends a (standard) non-interactive commitment to m_1.
2. The verifer $\mathcal{V}$ sends a random single bit challenge b.
3. If $b := 0$ then the prover sends f, r and the verifier checks if $\widehat{f}_{\mathrm{off}}(r)$ is computed correctly. Otherwise, $\mathcal{P}$ opens the commitment to m_1 and also sends $\widehat{f}_{\mathrm{on}}((m_1, \phi), r)$. $\mathcal{V}$ checks if the opening is valid and also runs $\mathrm{Dec}(\widehat{f}_{\mathrm{off}}(r), \widehat{f}_{\mathrm{on}}((m_1, \phi), r))$ and checks if this output is $(1, m_1, \phi)$.

In the case where $b = 0$, the commitment computationally hides m_1, whereas when $b = 1$, the privacy of the randomized encoding ensures that m_0 remains hidden. As shown in [21], this can indeed be formalized to prove that the protocol satisfies zero-knowledge. However, the protocol is only $1/2$ sound: in particular, a cheating prover can guess the verifier's challenge in advance, and use this to generate an accepting proof of a false statement.

Boosting Soundness. In order to boost soundness to close to 1, a natural idea is to parallel repeat this basic protocol to achieve negligible soundness error. But this idea does not work because we want a commit-and-prove: meaning that a malicious prover should be forced to commit to a single value and prove that it satisfies the predicate. In a naïve parallel repetition, a cheating prover could use different m's to compute the first message in different parallel repetitions. Therefore, we must find a way to ensure consistency of messages used across different parallel executions.

To achieve this, we augment the constant soundness protocol in the following way:

1. Instead of secret sharing m, the prover now secret shares $w := (m\|r)$ where r is a random element from a finite field $\mathbb{F}$.[7]. Let w_0 and w_1 be the secret shares. The prover constructs a function f that has w_0 hardwired in its description and takes as input the other share w_1 along with an augmented predicate ϕ' (which we will define later). f outputs $(1, w_1, \phi')$ if and only if $\phi'(w_0 \oplus w_1) = 1$. It constructs an offline encoding of this function $\widehat{f}_{\mathsf{off}}(r)$ and sends $\widehat{f}_{\mathsf{off}}(r)$ and also sends a (standard) non-interactive commitment to w_1.
2. The verifier chooses a random bit b as before and additionally chooses a random element $\alpha \leftarrow \mathbb{F} \setminus \{0\}$ and sends b, α to $\mathcal{P}$.
3. The prover computes $\gamma := r\alpha + m$ and sets the predicate $\phi'((m\|r))$ to check if $\phi(m) = 1$ and if γ is correctly computed. The prover sends γ and responds to the verifier's challenge bit as before.

In the parallel repetition of the above protocol, $\mathcal{P}$ chooses a "global" r that remains the same for each of the repetitions and also sends a single $\gamma := r\alpha + m$ in the third round; in each repetition, the predicate ϕ' shows that this "global" γ is consistent with the value w used in that repetition. We now show that the above augmented constant soundness protocol can force a prover to use consistent witness across multiple parallel executions. Say, the prover tries to use different witnesses $(r', m') \neq (r, m)$ across parallel repetitions. Then, with overwhelming probability $r\alpha + m \neq r'\alpha + m'$ by the Schwartz-Zippel lemma. Thus, the predicate that the prover is trying to prove in those repetitions is false and hence he will be caught if he tries to use inconsistent witness in many repetitions. We show that this parallel repetition satisfies witness indistinguishability.

[7] We assume that the message m also belongs to the same finite field.

Achieving Zero-Knowledge. As mentioned earlier, our three round parallel repetition of the augmented constant soundness protocol satisfies witness indistinguishability. Indeed, in order for it to achieve zero-knowledge property, it is necessary to have an additional round of interaction [11].

Our four round commit and prove ZK protocol follows the FLS paradigm [8] i.e., we run two special purpose witness indistinguishable protocols in opposite directions between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$. Recall that in the FLS paradigm the first protocol is a WI-PoK run by the verifier proving the knowledge of some trapdoor information. The second WI-PoK protocol run by the prover shows the knowledge of a witness for the statement x or the knowledge of verifier's trapdoor information. Intuitively, the soundness of the protocol follows from the security of the first WI-PoK and the zero-knowledgeness property follows from the observation that the simulator can rewind and extract the trapdoor information from the first WI-PoK and then use it in the second protocol.

In our construction, the first WI protocol run by $\mathcal{V}$ is a 3-round $\mathsf{two\text{-}com}$ protocol. Intuitively, the $\mathsf{two\text{-}com}$ protocol is a commitment to two random strings s_0 and s_1 such that commitment to s_b for a random $b \in \{0, 1\}$ is binding whereas the commitment to the other string s_{1-b} is equivocal. The trapdoor information is the string s_b. We demonstrate how to construct this primitive with black-box use of statistically binding commitment scheme using ideas from [34]. We additionally show that this trapdoor information can be extracted in expected polynomial time by rewinding the verifier. We wish to emphasize that the trapdoor that we use is in some sense "information theoretic" in nature and in contrast, the trapdoors usually used in the FLS paradigm are "crytographic" in nature such as the inverse of a given one-way function, or a signature under a public verification key, etc. Indeed, using such cryptographic trapdoors in the FLS paradigm leads to non-black-box use of one-way functions.

The second WI protocol run by $\mathcal{P}$ is essentially the 3-round WI protocol that we constructed earlier which proves that either the committed message m satisfies the predicate or m is the trapdoor. We show that a combination of these two special purpose WI protocols is a zero-knowledge commit and prove by carefully relying on the timing of the messages exchanged and the delayed input property of the second WI. We refer the reader to the main body for the details.

3 Preliminaries

In this section, we recall some preliminaries and tools that will be useful in our constructions. We will denote the security parameter by λ, and we will say that a function $f : \mathbb{N} \to \mathbb{N}$ is negligible if for every polynomial $p(\cdot)$ and all sufficiently large n it holds that $f(n) < \frac{1}{p(n)}$. We use the abbreviation PPT to denote probabilistic polynomial-time.

3.1 Commitment Schemes

A commitment scheme enables a party, known as the *sender* S, to commit to a value while keeping it secret from the *receiver* R – this property is called *hiding*. Furthermore, it is guaranteed that at a later stage, the opening of the commitment can only yield a single value – this property is called *binding*. We consider commitment schemes that are statistically binding and computationally hiding.

Definition 1 (Commitment schemes). *A commitment scheme $\langle C(m), R\rangle$ is a two-phase protocol between a committer $\mathcal{C}$ and receiver $\mathcal{R}$. At the beginning of the protocol, $\mathcal{C}$ obtains as input a message $m \in \{0,1\}^p$. Next, $\mathcal{C}$ and $\mathcal{R}$ execute the commit phase, and obtain commitment transcript $\tau \leftarrow \mathsf{Commit}\langle\mathcal{C}(m), \mathcal{R}\rangle$. They also store (private) randomness used respectively by $\mathcal{C}$ and $\mathcal{R}$ as $\mathsf{state}_{\mathcal{C},\tau}$ and $\mathsf{state}_{\mathcal{R},\tau}$. At the end of this phase, $\mathcal{R}$ outputs 0 or 1, where 1 denotes that $\mathcal{R}$ accepted the commitment phase. The view of the receiver (including its coins, any auxiliary information z and transcript) at the end of this phase is denoted by $\mathsf{View}_{\mathcal{R}}\langle\mathcal{C}(M), \mathcal{R}(z)\rangle$.*

Later, $\mathcal{C}$ and $\mathcal{R}$ possibly engage in another (interactive) decommit phase, which we denote by $\mathsf{Decommit}\langle\tau, \mathcal{C}(m, \mathsf{state}_{\mathcal{C},\tau}), \mathcal{R}(\mathsf{state}_{\mathcal{R},\tau})\rangle$ at the end of which $\mathcal{R}$ outputs $\perp$ or a message $\widetilde{m} \in \{0,1\}^p$.

We require these algorithms to satisfy the following properties:

- **Correctness:** *If $\mathcal{C}, \mathcal{R}$ honestly follow the protocol,* $\Pr[\mathcal{R}$ *accepts the decommitment*$] = 1 - \mathsf{negl}(\lambda)$.
- **Computational hiding.** *For every PPT machine R^* with auxiliary information z, the distributions $\{\mathsf{View}_{\mathcal{R}}\langle\mathcal{C}(m), \mathcal{R}(z)\rangle\}$ and $\{\mathsf{View}_{\mathcal{R}}\langle\mathcal{C}(0), \mathcal{R}(z)\rangle\}$ are computationally indistinguishable.*
- **Statistical binding.** *For any (unbounded) malicious $\mathcal{C}^*$,*

  $$\Pr\left[\mathcal{R} \text{ accepts decommitment to } \widetilde{m}_1 \text{ and } \widetilde{m}_2 \text{ where } \widetilde{m}_1 \neq \widetilde{m}_2\right] \leq \mathsf{negl}(\lambda)$$

  where the probability is over the randomness of sampling $(\tau \leftarrow \mathsf{Commit}\langle\mathcal{C}^*, \mathcal{R}\rangle)$, $(\widetilde{m}_1 \leftarrow \mathsf{Decommit}\langle\tau, \mathcal{C}^*, \mathcal{R}(\mathsf{state}_{\mathcal{R},\tau})\rangle)$ and $(\widetilde{m}_2 \leftarrow \mathsf{Decommit}\langle\tau, \mathcal{C}^*, \mathcal{R}(\mathsf{state}_{\mathcal{R},\tau})\rangle)$. We will say that the scheme satisfies computational binding if the above holds for any PPT committer $\mathcal{C}^*$ with auxiliary input z.

We now define an extractable commitment scheme [37, 38]. Intuitively, a commitment scheme is extractable if there exists an expected polynomial time machine that can extract the value committed by a cheating committer.

Definition 2 (Extractable Commitments). *A commitment scheme is said to be extractable, if there exists a PPT oracle algorithm E that given $\tau \leftarrow \mathsf{Commit}\langle\mathcal{C}^*, \mathcal{R}\rangle$ and oracle access to $\mathcal{C}^*$, outputs $\widetilde{m}, r$ such that $\exists r$ where $\tau = \mathsf{Commit}\langle\mathcal{C}(\widetilde{m}), \mathcal{R}\rangle$ using randomness r for $\mathcal{C}$.*

An equivocal commitment scheme allows an expected polynomial time machine called as the *equivocator* to equivocate a commitment transcript to any chosen committed value. Equivocal commitments have been extensively used to obtain round optimal constructions of secure two-party and multiparty computations [29, 10].

Definition 3 (Equivocal Commitments). *A commit-and-prove scheme is equivocal if there exists a PPT oracle algorithm* Eq *that interacts with oracle access to any malicious receiver* $\mathcal{R}^*$ *to output a commitment transcript* $\widetilde{\tau}$*. Next, it obtains externally generated string* m'*, and then runs* $\mathsf{Decommit}\langle \widetilde{\tau}, \mathsf{Eq}^{\mathcal{R}^*}, \mathcal{R}^* \rangle$*. Then, we require that the distributions*

$$\mathsf{View}_{\mathcal{R}^*}(\tau \leftarrow \mathsf{Commit}\langle \mathcal{C}(M'), \mathcal{R}^* \rangle, \mathsf{Decommit}\langle \tau, \mathcal{C}, \mathcal{R}^* \rangle) \text{ and}$$

$$\mathsf{View}_{\mathcal{R}^*}(\widetilde{\tau} \leftarrow \mathsf{Commit}\langle \mathsf{Eq}^{\mathcal{R}^*}, \mathcal{R}^* \rangle, \mathsf{Decommit}\langle \widetilde{\tau}, \mathsf{Eq}^{\mathcal{R}^*}, \mathcal{R}^* \rangle)$$

are computationally indistinguishable.

3.2 Commit-and-Prove Protocols

We start with the definition of commit and prove witness indistinguishable proof of knowledge. Our construction of commit and prove witness indistinguishable proof of knowledge satisfies a weaker notion of 1-of-2 binding. Intuitively, 1-of-2 binding states that there exists at most two different messages that a committed transcript can be opened to.

Definition 4 (Commit-and-Prove Witness Indistinguishable Proof of Knowledge). *A commit-and-prove witness indistinguishable proof of knowledge is a protocol between a prover* $\mathcal{P}$ *and verifier* $\mathcal{V}$*. It consists of two phases, a commit phase and reveal phase.*

In the commit phase, $\mathcal{P}$ *interacts with* $\mathcal{V}$ *to commit to a message* m*. It also proves that the* m *satisfies some predicate* ϕ*, in other words it proves that* $\phi(m) = 1$*. Let* τ *denote the transcript* $\tau \leftarrow \mathsf{Commit\text{-}and\text{-}Prove}\langle \mathcal{P}(m, \phi), \mathcal{V}(\phi) \rangle$*. They also store (private) randomness used respectively by* $\mathcal{P}$ *and* $\mathcal{V}$ *as* $\mathsf{state}_{\mathcal{P},\tau}$ *and* $\mathsf{state}_{\mathcal{V},\tau}$*. At the end of this phase,* $\mathcal{V}$ *outputs* 0 *or* 1*, where* 1 *denotes that* $\mathcal{V}$ *accepted the commit-and-prove phase.*

Later, the parties $\mathcal{P}$ *and* $\mathcal{V}$ *possibly engage in another decommit phase, which we denote by* $\mathsf{Decommit}\langle \tau, \mathcal{P}(m, \mathsf{state}_{\mathcal{P},\tau}), \mathcal{V}(\mathsf{state}_{\mathcal{V},\tau}) \rangle$*, at the end of which* $\mathcal{V}$ *outputs* $\perp$ *or* $\widetilde{m} \in \{0,1\}^p$*.*

We require the protocol to satisfy the following conditions:

- **Completeness:** *If* $\mathcal{P}, \mathcal{V}$ *honestly follow the protocol,* $\Pr[\mathcal{V} \text{ accepts the proof}] = 1 - \mathsf{negl}(\lambda)$*.*
- **Witness Indistinguishability.** *Let the view of a malicious verifier* $\mathcal{V}^*$ *at the end of the commit phase when the honest prover has input message* m *be denoted by* $\mathsf{View}_{\mathcal{V}^*}(\mathsf{Commit\text{-}and\text{-}Prove}\langle \mathcal{P}(m), \mathcal{V}^*(z) \rangle)$*. For any malicious verifier* V^**, and any two messages* m_1, m_2 *such that* $\phi(m_1) = 1$ *and* $\phi(m_2) = 1$*, the distributions* $\{P(m_1), V^*(z)\}$ *and* $\{P(m_2), V^*(z)\}$ *are computationally indistinguishable.*

– **Proof of Knowledge.** *There exists a PPT oracle algorithm E that given oracle access to $\mathcal{P}^*$ and $\tau \leftarrow \mathsf{Commit\text{-}and\text{-}Prove}\langle \mathcal{P}^*, \mathcal{R}(\phi)\rangle$ outputs $\widetilde{m}$ such that the following properties are satisfied for every PPT $\mathcal{P}^*$:*
  - $\phi(\widetilde{m}) = 1$.
  - **1-of-2-Binding.** *This requires that the committer cannot decommit to two values m_1, m_2, both of which are different from $\widetilde{m}$. In other words, we require the commit-and-prove to bind any malicious committer to at least one out of two values. Formally,* $\Pr[\widetilde{m} \notin \{m_1, m_2\}] \leq \mathsf{negl}(\lambda)$, *whenever* $m_1 \leftarrow \mathsf{Decommit}\langle \tau, \mathcal{P}^*, \mathcal{V}(\mathsf{state}_{\mathcal{V},\tau})\rangle$, *and also when* $m_2 \leftarrow \mathsf{Decommit}\langle \tau, \mathcal{P}^*, \mathcal{V}(\mathsf{state}_{\mathcal{V},\tau})\rangle$.

We now give the definition of commit and prove zero-knowledge argument of knowledge. We include the equivocality property into our zero-knowledge condition. This will be helpful when proving the security of our construction of equivocal commitment scheme.

Definition 5 (Commit-and-Prove Zero-Knowledge Arguments of Knowledge). *A commit-and-prove zero-knowledge argument of knowledge is a protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$. It consists of two phases, a commit phase and reveal phase.*

In the commit phase, $\mathcal{P}$ interacts with $\mathcal{V}$ to commit to a message m. It also proves that the m satisfies some predicate ϕ, in other words it proves that $\phi(m) = 1$. Let τ denote the transcript $\tau \leftarrow \mathsf{Commit\text{-}and\text{-}Prove}\langle \mathcal{P}(m, \phi), \mathcal{V}(\phi)\rangle$. They also store (private) randomness used respectively by $\mathcal{P}$ and $\mathcal{V}$ as $\mathsf{state}_{\mathcal{P},\tau}$ and $\mathsf{state}_{\mathcal{V},\tau}$. At the end of this phase, $\mathcal{V}$ outputs 0 or 1, where 1 denotes that $\mathcal{V}$ accepted the commit-and-prove phase.

Later, the parties $\mathcal{P}$ and $\mathcal{V}$ possibly engage in another decommit phase, which we denote by $\mathsf{Decommit}\langle \tau, \mathcal{P}(m, \mathsf{state}_{\mathcal{P},\tau}), \mathcal{V}(\mathsf{state}_{\mathcal{V},\tau})\rangle$, at the end of which $\mathcal{V}$ outputs $\perp$ or $\widetilde{m} \in \{0,1\}^p$.

We require the protocol to satisfy the following conditions:

– **Completeness:** *If $\mathcal{P}, \mathcal{V}$ honestly follow the protocol,* $\Pr[\mathcal{V}$ *accepts the proof*$] = 1 - \mathsf{negl}(\lambda)$.
– **Argument of Knowledge.** *There exists a PPT oracle algorithm E that given oracle access to $\mathcal{P}^*$ and $\tau \leftarrow \mathsf{Commit\text{-}and\text{-}Prove}\langle \mathcal{P}^*, \mathcal{R}(\phi)\rangle$ outputs $\widetilde{m}$ such that the following properties are satisfied for every PPT $\mathcal{P}^*$:*
  - $\phi(\widetilde{m}) = 1$.
  - **Computational Binding.** $\Pr\left[m \leftarrow \mathsf{Decommit}\langle \tau, \mathcal{P}^*, \mathcal{R}(\mathsf{state}_{\mathcal{R},\tau})\rangle \wedge m \neq \widetilde{m}\right] \leq \mathsf{negl}(\lambda)$.
– **Zero-Knowledge.** *Let $\mathsf{View}_{\mathcal{V}^*}(\mathsf{Commit\text{-}and\text{-}Prove}\langle \mathcal{P}(m), \mathcal{V}^*(z)\rangle)$ denote the view of a malicious verifier $\mathcal{V}^*$ at the end of the commit phase when the honest prover has input message m such that $\phi(m) = 1$. There exists a simulator Sim that outputs $\mathsf{View}_{\mathsf{Commit}}(\mathsf{Sim}^{\mathcal{V}^*})$. Next, it obtains input m and outputs $\mathsf{View}_{\mathsf{Decommit}}(\mathsf{Sim}^{\mathcal{V}^*}(m))$. Then, we require that for all values m,*

$$\Big(\mathsf{View}_{\mathsf{Commit}}(\mathsf{Sim}^{\mathcal{V}^*}), \mathsf{View}_{\mathsf{Decommit}}(\mathsf{Sim}^{\mathcal{V}^*}(m))\Big) \overset{c}{\approx}$$
$$\Big(\mathsf{View}_{\mathcal{V}^*}(\mathsf{Commit\text{-}and\text{-}Prove}\langle \mathcal{P}(m), \mathcal{V}^*(z)\rangle), \mathsf{Decommit}\langle \tau, \mathcal{P}(\mathsf{state}_{\mathcal{P},\tau}), \mathcal{V}^*\Big)$$

In particular, this also implies that any malicious verifier V^*, and any two messages m_1, m_2 such that $R(\phi, m_1) = 1$ and $R(\phi, m_2) = 1$, the distribution $\mathsf{View}_{\mathcal{V}^*}(\mathsf{Commit\text{-}and\text{-}Prove}\langle \mathcal{P}(m_1), \mathcal{V}^*(z)\rangle)$ *and the distribution* $\mathsf{View}_{\mathcal{V}^*}(\mathsf{Commit\text{-}and\text{-}Prove}\langle \mathcal{P}(m_2), \mathcal{V}^*(z)\rangle)$ *are computationally indistinguishable.*

Remark 1. A commit-and-prove protocol is said to satisfy delayed-input completeness, if $\mathcal{P}, \mathcal{V}$ obtain the predicate ϕ in the last round of a protocol.

3.3 Robust Offline/Online Randomized Encoding

We start with the definition of a randomized encoding [24, 2, 3].

Definition 6 (Randomized Encoding). *Let $f : \{0,1\}^n \to \{0,1\}^\ell$ be a function. Then a function $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ is said to be a randomized encoding of f, if:*

- **Correctness:** *There exists a decoder algorithm* Dec *such that for any input $x \in \{0,1\}^n$, except with negligible probability over the randomness of the encoding and the random coins of* Dec, *it holds that* $\mathsf{Dec}(\hat{f}(x, U_m)) = f(x)$.
- **Computational (statistical) privacy:** *There exists a PPT simulator $\mathcal{S}$, such that for any input $x \in \{0,1\}^n$ the following distributions are computationally (statistically) indistinguishable:*
  - $\{\hat{f}(x, U_m)\}_{n\in\mathbb{N}, x\in\{0,1\}^n}$, *and,*
  - $\{\mathcal{S}(f(x))\}_{n\in\mathbb{N}, x\in\{0,1\}^n}$

We recall the definition of robust randomized encoding from [21].

Definition 7 (Robust Offline/Online Randomized Encoding). *[21] A randomized encoding is called an online/offline encoding, if there exists functions $\widehat{f}_{\mathrm{off}}$ and $\widehat{f}_{\mathrm{on}}$ such that $\widehat{f}(x; r) = (\widehat{f}_{\mathrm{off}}(r), \widehat{f}_{\mathrm{on}}(x, r))$. That is, there exists an offline component that does not depend on the input, and an online component which is a function of the input. It is called robust if additionally, it holds that: if there exists no x such that $f(x) = a$, then for any r, there does not exist any z such that* $\mathsf{Dec}(\widehat{f}_{\mathrm{off}}(r), z) = a$.

In Appendix A, we describe a simplified variant of the construction of robust offline/online randomized encodings from [21], that only assumes one-way functions.

4 Three-round Black-box Commit-and-Prove WIPoK

In this section, we describe a three-round black-box commit-and-prove witness indistinguishable proof of knowledge protocol.

Input: The prover $\mathcal{P}$ and verifier $\mathcal{V}$ have common input a predicate ϕ. $\mathcal{P}$ additionally obtains input $m \in \mathbb{F}$ (where $\mathbb{F}$ is a finite field) such that $\phi(m) = 1$.

Definition of f: The function f has $w_0 \in \mathbb{F}^3$ hardwired. It takes as input the predicate ϕ, two elements $\alpha, \beta \in \mathbb{F}$, an element $w_1 \in \mathbb{F}^3$, two values $(a, b) \in \mathbb{F}$, and does the following:

1. Set $(m\|r\|s) = w_0 + w_1$.
2. If $a \neq (r + \alpha m)$ and $b \neq (s + \beta m)$ output $\perp$.
   Else, output $(\phi(m), \phi, \alpha, \beta, w_1, (a, b))$.

Protocol:

1. $\mathcal{P}$ chooses $r \xleftarrow{\$} \mathbb{F}$. For $i \in [\lambda]$, $\mathcal{P}$ does the following:
   (a) Choose $w_0^i \xleftarrow{\$} \mathbb{F}^3$ and compute $w_1^i := w_0^i \oplus (m\|r\|\perp)$.
   (b) Choose $\omega_i \xleftarrow{\$} \{0,1\}^*$ and compute $\widehat{f}^{\text{off}}(\omega_i)$ and $\tau_i \leftarrow \mathsf{Com}(w_1^i)$.
   (c) Send $(\widehat{f}^{\text{off}}(\omega_i), \tau_i)$.
2. $\mathcal{V}$ sends $\mathsf{ch} \xleftarrow{\$} \{0,1\}^\lambda$, $\alpha \xleftarrow{\$} \mathbb{F}$ and $\beta \leftarrow \mathbb{F}$.
3. $\mathcal{P}$ chooses $b \xleftarrow{\$} \mathbb{F}$, and sets $a = (r + \alpha m)$. It sends a, b. Additionally, for every $i \in [\lambda]$, $\mathcal{P}$ does the following:
   (a) If $\mathsf{ch}_i = 0$, send w_0^i, ω_i.
   (b) If $\mathsf{ch}_i = 1$, send $\widehat{f}^{\text{on}}(\phi||\alpha||\beta||w_1^i\|(a, b), \omega_i)$, and decommit to τ_i.

Verification Phase:

– Output 1 if the following checks pass for every $i \in [\lambda]$:
  1. If $\mathsf{ch}_i = 0$, check if the received w_0^i is embedded in the circuit $\widehat{f}^{\text{off}}$ computed using randomness r_i.
  2. If $\mathsf{ch}_i = 1$, run $\text{Dec}(\widehat{f}^{\text{off}}, \widehat{f}^{\text{on}})$ and accept if the evaluation outputs $(1, \phi, \alpha, \beta, w_1^i, (a, b))$ where w_1^i is obtained from the decommitment to τ_i.

Fig. 1. Black-Box Witness Indistinguishable Proof of Knowledge

4.1 Construction

The construction is described in Figure 1, and uses a robust randomized encoding $(\widehat{f}^{\text{off}}, \widehat{f}^{\text{on}})$, secure in the presence of adaptive choice of inputs.

We have the following theorem.

Theorem 1. *The protocol described in Figure 1 is a black-box commit-and-prove witness indistinguishable argument of knowledge according to Definition 4.*

The completeness of the protocol can be easily verified from inspection. Furthermore, the protocol only makes black-box access to a non-interactive commitment scheme and a robust randomized encoding. Recall that a robust randomized encoding can be constructed from black-use of a one-way function. We now show witness indistinguishability.

Lemma 1. *The protocol described in Figure 1 satisfies witness indistinguishability according to Definition 4.*

Proof. To prove witness indistinguishability of this protocol, we consider the following sequence of hybrid experiments.

The first hybrid Hyb_0 corresponds to the view of a (malicious) verifier interacting with an honest prover that follows the protocol in Figure 1 using the message m_0 for the predicate ϕ.

We define a hybrid $\mathsf{Hyb}_{0,k}$ for each $k \in [0, \lambda]$ that corresponds to the view of a (malicious) verifier interacting with a prover that uses the message m_1 in the first k instances and the message m_0 in the remaining instances. To be more precise, the prover in $\mathsf{Hyb}_{0,k}$ does the following:

- In round-1,
  1. $\mathcal{P}$ chooses $r, s \xleftarrow{\$} \mathbb{F}$.
  2. For $i \in [k]$, $\mathcal{P}$ does the following:
     (a) Choose $w_0^i \xleftarrow{\$} \mathbb{F}^3$ and compute $w_1^i := w_0^i \oplus (m_1 \| \bot \| s)$.
     (b) Choose $\omega_i \xleftarrow{\$} \{0,1\}^*$ and compute $\widehat{f}^{\text{off}}(\omega_i)$ and $\tau_i \leftarrow \mathsf{Com}(w_1^i)$.
     (c) Send $(\widehat{f}^{\text{off}}(\omega_i), \tau_i)$.
  3. For $i \in [k+1, \lambda]$, $\mathcal{P}$ does the following:
     (a) Choose $w_0^i \xleftarrow{\$} \mathbb{F}^3$ and compute $w_1^i := w_0^i \oplus (m_0 \| r \| \bot)$.
     (b) Choose $\omega_i \xleftarrow{\$} \{0,1\}^*$ and compute $\widehat{f}^{\text{off}}(\omega_i)$ and $\tau_i \leftarrow \mathsf{Com}(w_1^i)$.
     (c) Send $(\widehat{f}^{\text{off}}(\omega_i), \tau_i)$.
- In round-3,
  1. $\mathcal{P}$ sends $b := s + \beta m_1$ and $a = r + \alpha m_0$. Additionally, for every $i \in [\lambda]$, $\mathcal{P}$ does the following:
     (a) If $\mathsf{ch}_i = 0$, send w_0^i, ω_i.
     (b) If $\mathsf{ch}_i = 1$, send $\widehat{f}^{\text{on}}(\phi \| \alpha \| \beta \| w_1^i \| (a, b), \omega_i)$, and decommit to τ_i.

Claim. Assuming the hiding property of Com and the adaptive security of robust randomized encoding, $\mathsf{Hyb}_{0,k-1} \overset{c}{\approx} \mathsf{Hyb}_{0,k}$ for each $k \in [\lambda]$.

Proof. Assume for the sake of contradiction that there exists a malicious verifier that can distinguish $\mathsf{Hyb}_{0,k-1}$ and $\mathsf{Hyb}_{0,k}$ with non-negligible probability. We will construct an adversary $\mathcal{B}$ that breaks the security of either the robust randomized encoding or the hiding property of Com with non-negligible probability. $\mathcal{B}$ chooses a bit $b_k \xleftarrow{\$} \{0,1\}$.

Case-1: $b_k = 0$**.** In this case, $\mathcal{B}$ does the following:

1. For all $i \neq k$, $\mathcal{B}$ generates the commitments and the randomized encoding as in $\mathsf{Hyb}_{0,k-1}$. For $i = k$, it does the following:
   (a) Choose $w_0^i \xleftarrow{\$} \mathbb{F}^3$ and compute $w_1^i := w_0^i \oplus (m_0 \| r \| \bot)$ and $\widehat{w}_1^i := w_0^i \oplus (m_1 \| \bot \| s)$. Give the two messages $w_0^i, \widehat{w}_1^i$ as the challenge messages to the hiding property of Com. Obtain the challenge commitment τ_i.

   (b) Choose $\omega_i \xleftarrow{\$} \{0,1\}^*$ and compute $\widehat{f}^{\rm off}(\omega_i)$ and $\tau_i \leftarrow \mathsf{Com}(w_1^i)$.
   (c) Send $(\widehat{f}^{\rm off}(\omega_i), \tau_i)$.
2. If ch_k obtained from the receiver is not equal to b_k, we abort and output a random bit. Else, continue the protocol as per the description of $\mathsf{Hyb}_{0,k-1}$. Output whatever the verifier outputs
3. Note that if τ_k is a commitment to w_1^i then the distribution is identical to $\mathsf{Hyb}_{0,k-1}$; else, it is identical to distribution $\mathsf{Hyb}_{0,k}$. Thus, if the malicious verifier can distinguish between $\mathsf{Hyb}_{0,k-1}$ and $\mathsf{Hyb}_{0,k}$ with probability p then $\mathcal{B}$ breaks the hiding of the commitment scheme with probability at least $p/2$.

Case-2: $b_k = 1$**.** In this case, $\mathcal{B}$ does the following:

1. For all $i \neq k$, $\mathcal{B}$ generates the commitments and the randomized encoding as in $\mathsf{Hyb}_{0,k-1}$. For $i = k$, it does the following:
   (a) Choose $w_1^i \xleftarrow{\$} \mathbb{F}^3$ and computes $w_0^i := w_1^i \oplus (m_0 \| r \| s)$ and $\widehat{w}_0^i := w_1^i \oplus (m_1 \| r \| s)$. Give to the randomized encoding challenger two circuits $f[w_0^i]$ and $f[\widehat{w}_0^i]$ as the challenge circuits. Obtain $\widehat{f}^{\rm off}$ as the challenge circuit.
   (b) Send $\widehat{f}^{\rm off}$ and $\tau_i \leftarrow \mathsf{Com}(w_1^i)$.
2. If ch_k obtained from the receiver is not equal to b_k, we abort and output a random bit. Else,
   (a) Obtain α from the verifier.
   (b) Send $\phi||\alpha||\beta||w_1^i\|(a, b), \omega_i$ as the challenge input to the randomized encoding challenger and obtain $\widehat{f}^{\rm on}$.
   (c) Send $\widehat{f}^{\rm on}$ as response to ch_k and decommit to τ_k.
3. Finally, output whatever the verifier outputs.

Notice that the output of the two circuits $f[w_0^i]$ and $f[\widehat{w}_0^i]$ on the challenge input is exactly the same. Thus, $\mathcal{B}$ constitutes a valid challenger to the adaptive security of randomized encoding. Thus, if $\widehat{f}^{\rm off}$ corresponds to a offline randomized encoding of $f_{w_0^i}$, the view of the malicious verifier is identical to $\mathsf{Hyb}_{0,k-1}$. Else, the view is identical to $\mathsf{Hyb}_{0,k}$. Thus, a malicious verifier distinguishing $\mathsf{Hyb}_{0,k-1}$ and $\mathsf{Hyb}_{0,k}$ can be used to break the security of robust randomized encodings.

We now prove that $\mathsf{Hyb}_{0,0}$ is identically distributed to Hyb_0. Notice that $\mathsf{Hyb}_{0,0}$ is the same as Hyb_0, except that the prover sends $b = s + \beta m_1$ instead of sampling b uniformly at random. Since s is information theoretically hidden in both $\mathsf{Hyb}_{0,0}$ and Hyb_0 it follows that both these distributions are identical. A similar argument shows that $\mathsf{Hyb}_{0,\lambda}$ is identical to Hyb_1. This completes the proof of the claim.

The proof of WI follows by noting that $\mathsf{Hyb}_{0,\lambda}$ is distributed identically to the case where the prover uses the witness W_1 to generate the proof.

We will now prove that it is an argument of knowledge:

Lemma 2. *The protocol in Figure 1 is a proof of knowledge against PPT provers, even for statements chosen adaptively by such a prover in the last round, according to Definition 4.*

Proof. We begin by describing the extractor (having oracle access to a PPT prover $\mathcal{P}^*$) that takes as input an accepted transcript $\mathbb{T}$ and outputs a value $\widetilde{m}$ such that $\phi(\widetilde{m}) = 1$ and there exists at most two messages $\widetilde{m}_1, \widetilde{m}_2$ such that $\widetilde{m} \in \{\widetilde{m}_1, \widetilde{m}_2\}$ and either $\mathcal{P}^*$ will decommit to $\widetilde{m}_1$ or $\widetilde{m}_2$.

The extractor rewinds the cheating prover $\mathcal{P}^*$ to the beginning of the third round multiple times, and gives different uniformly chosen challenge messages $ch \overset{\$}{\leftarrow} \{0,1\}^\lambda$. It stops when it obtains for some $i \in [\lambda]$, two decommitments w_0^i, w_1^i such that $w_0^i \oplus w_1^i = (m\|r\|s)$, and outputs m if $\phi(m)$, and $r + \alpha m = a$ or $s + \beta m = b$ from the main thread.

Let $\mathbb{T}$ be the accepted protocol transcript. Because of robustness of the randomized encoding and a simple averaging argument, we note that with overwhelming probability over the choice of random challenge $ch \in \mathbb{T}$, at least $\lambda - O(\log^2 \lambda)$ indices i are such that $f_{w_0^i}(\phi, \alpha, \beta, w_1^i, a, b) \neq \bot$. Let S be the set of indices i such that the above is true. Then, we have for each $i \in S$, let $w_1^i \oplus w_0^i = (m_i\|r_i\|s_i)$ where $\phi(m_i) = 1$. Further, for every $i \in S$, we now have from the definition of f that, $r_i + \alpha m_i = a$ and $s_i + \beta m_i = b$. With overwhelming probability over α, β, this is possible only if there exists at most two values $\widetilde{m}_1, \widetilde{m}_2$ such that $m_i \in \{\widetilde{m}_1, \widetilde{m}_2\}$ for all $i \in S$ (by Schwartz-Zippel lemma).

We finally argue that the extractor runs in expected polynomial time. Let p be the probability that conditioned on the first two messages of the protocol, the prover $\mathcal{P}^*$ generates an accepting proof. Since the running time of the extractor in each rewind is bounded by some polynomial $\mathrm{poly}(\lambda)$, we have that the expected running time of the extractor is $p.\frac{\mathrm{poly}(\lambda)}{p} = \mathrm{poly}(\lambda)$.

4.2 Black-Box One-Binding Commitment to Two Strings

In this section, we describe how to use the black-box commit-and-prove WIPoK to generate a commitment to two strings such that one of the two commitments is binding, and the other can be freely equivocated by a simulator. Such a protocol can also be built using ideas from [34], however, we give a direct instantiation via a slight modification of our black-box commit-and-prove WIAoK. This scheme is referred to as two-com, and is described in Figure 2.

We also note that unlike [34], when we use scheme two-com, honest parties will never need to rely on equivocation, and equivocation will only be used in the proof of security.

Witness indistinguishability of the argument of binding of one of the two commitments follows directly via witness indistinguishability of the underlying protocol, using an identical proof to the one in Lemma 1.

We will now argue why the protocol in Figure 2 is such that any (malicious) committer is bound to one of the two openings $m \in \{m_0, m_1\}$, by the end of the first round. This relies on soundness of the witness indistinguishable argument. Specifically, by the Schwartz-Zippel lemma, there exist at most two witnesses W, W' such that at least $(\lambda - \log^2 n)$ parallel executions generated by a malicious committer, have a commitment to either W or W', or both. Now, because of the soundness of individual WI arguments, once the first message has been

Input: Committer $\mathcal{C}$ has input two messages m_0, m_1.

Definition of f: The function f has $w_0 \in \mathbb{F}^2$ hardwired. *The relation* $R(x, w) = 1$ *if and only if* ($x_1 = w$ *OR* $x_2 = w$)*, where* $x = x_1||x_2$.
The function f takes as input an instances $x \in \mathbb{F}^2$, an element $\alpha \in \mathbb{F}$, an element $w_1 \in \mathbb{F}^2$, two values $(a, \widetilde{a}) \in \mathbb{F}^2$, and does the following:

1. Compute $(w\|r\|\widetilde{r}) = w_0 + w_1$.
2. If $a \neq (r + \alpha w)$ and $\widetilde{a} \neq (\widetilde{r} + \alpha_1 w)$, output $\perp$. Else, output $(R(x, w), x, \alpha, w_1, (a, \widetilde{a}))$.

Protocol:

1. $\mathcal{C}$ chooses $r \xleftarrow{\$} \mathbb{F}$. For $i \in [\lambda]$, $\mathcal{P}$ does the following with $w = m_0$:
   (a) Chooses $w_0^i \xleftarrow{\$} \mathbb{F}$ and computes $w_1^i := w_0^i \oplus (w\|r)$.
   (b) Chooses $r_i \xleftarrow{\$} \{0,1\}^*$ and computes $\widehat{f}^{\text{off}}_{w_0^i}(r_i)$, $\sigma_i \leftarrow \mathsf{Com}(r_i)$ and $\tau_i \leftarrow \mathsf{Com}(w_1^i)$.
   (c) Sends $(\widehat{f}^{\text{off}}_{w_0^i}(r_i), \sigma_i, \tau_i)$.
2. $\mathcal{R}$ sends $\mathsf{ch} \xleftarrow{\$} \{0,1\}^\lambda$ and $\alpha \xleftarrow{\$} \mathbb{F}$.
3. $\mathcal{P}$ sends the opening $x = (m_0||m_1)$. Additionally, $\mathcal{P}$ chooses $\widetilde{a} \xleftarrow{\$} \mathbb{F}$ sends $\widetilde{a}$ and $a = r + \alpha w$. Additionally, for every $i \in [\lambda]$, $\mathcal{P}$ does the following:
   (a) If $\mathsf{ch}_i = 0$, sends w_0^i, s_0^i and decommits to σ_i.
   (b) If $\mathsf{ch}_i = 1$, sends $\widehat{f}^{\text{on}}_{w_0^i}(x||\alpha||w_1^i\|(a, \widetilde{a}); r_i)$, and decommits to τ_i.

Verification Phase:

– Output 1 if the following checks pass for every $i \in [\lambda]$:
  1. If $\mathsf{ch}_i = 0$, check if the received w_0^i is embedded in the circuit $\widehat{f}^{\text{off}}_{w_0^i}$ computed using randomness r_i. Check that the decommitment information to σ_i is correct.
  2. If $\mathsf{ch}_i = 1$, run the evaluation algorithm for randomized encoding $\widehat{f}^{\text{off}}_{w_0^i}$ with $\widehat{f}^{\text{on}}_{w_0^i}$ as input and accept if the evaluation outputs $(1, x, \alpha, w_1^i, (a, \widetilde{a}))$ where w_1^i is obtained from the decommitment to τ_i.

Fig. 2. Commitment to Two Strings where One is Binding

committed, in the third message, any (malicious) committer can only open to m_0, m_1 such that:

– If in the first message, all but $\log^2 n$ commitments were to the same witness W, then either $W = m_0$ or $W = m_1$.
– If in the first message, all but $\log^2 n$ commitments were to two witnesses W, W' then $W = m_0, W' = m_1$ or vice-versa.

5 Four-round Black-box Commit-and-Prove Zero-Knowledge

In this section, we describe a black-box commit-and-prove zero-knowledge argument in four rounds based on injective one-way functions. We start with a description of the main tools used in the construction.

5.1 Construction

Our construction is described formally in Figure 3, and makes use of the following primitives:

- A non-interactive, statistically binding commitment Com.
- A three-round commitment to two strings, together with a black-box witness-indistinguishable proof that one of the two commitments is binding by the end of the first round. We also require the other commitment to be equivocal. Such a scheme is described in [34], Section 3. Let $\mathsf{two\text{-}com}(s_1, s_2)$ denote such a scheme for committing to strings s_1 and s_2.
- A robust randomized encoding $\widehat{f}^{\mathrm{off}}, \widehat{f}^{\mathrm{on}}$ according to Definition 6.

5.2 Proof of Security

We start with the lemma which shows that the protocol described in Figure 1 is a commitment to the witness w.

Lemma 3. *The protocol described in Figure 3 is a statistically binding commitment to the element $w \in \mathbb{F}$.*

Proof. We start with the description of the decommit phase and then argue statistical binding and computational hiding of the protocol.

The decommit phase involves opening the commitments σ_i, σ'_i and σ^*_i and sending w^i_0 for every $i \in [\lambda]$. For each $i \in [\lambda]$, compute $w_i \| r_i := w^i_0 + w^i_1$ (where a value is substituted with a default symbol if the decommitment information is not valid) and output the value w that occurs in more than $\lambda/2$ positions. If there is no w that occurs in more that $\lambda/2$ positions then we reject the decommitment information.

Since the commitment sent in the second round of the protocol is statistically binding and we have defined the decommitment phase to output the majority of the committed values, we note that there can exist at most one valid decommitment to a protocol transcript except with negligible probability. Thus, the protocol is statistically binding. We note that computational hiding property follows the zero-knowledge property we later show.

Lemma 4. *The protocol in Figure 3 is an argument of knowledge against PPT provers, even for statements chosen adaptively by such a prover in the last round, according to Definition 5.*

Input: The prover $\mathcal{P}$ and verifier $\mathcal{V}$ have common input x and relation R. $\mathcal{P}$ additionally obtains input w such that $R(x, w) = 1$. We assume that $w \in \mathbb{F}$ where $\mathbb{F}$ is a finite field.

Definition of f: The function f has hardwired a share of the witness w_0, and a share of trapdoor information s_0. It takes as input the instance x, a challenge α, a share of the witness w_1, a share of trapdoor information s_1, a value a, $\widehat{s}_0, \widehat{s}_1$ (recovered from the third message of the $\mathsf{two\text{-}com}$) and does the following:

1. Compute $s = s_0 \oplus s_1$. If $\widehat{s}_0 = s$ or $\widehat{s}_1 = s$, output $(1, x, w_1, s_1, a)$. Else, continue.
2. Compute $w \| r = w_0 \oplus w_1$. If $a \neq (r\alpha + w)$, output $\perp$. Else, output $(R(x, w), x, w_1, s_1, a)$.

Protocol:

1. $\mathcal{V}$ picks strings $\widehat{s}_0, \widehat{s}_1 \xleftarrow{\$} \{0,1\}^{2\lambda}$ and sends the first message π_1 of $\mathsf{two\text{-}com}(\widehat{s}_0, \widehat{s}_1)$.
2. $\mathcal{P}$ chooses $r \xleftarrow{\$} \mathbb{F}$, and $s \leftarrow \{0,1\}^\lambda$. For $i \in [\lambda]$ it does the following:
   (a) Choose w_0^i uniformly at random and compute $w_1^i := w_0^i \oplus (w \| r)$.
   (b) Choose s_0^i uniformly at random and compute $s_1^i := s_0^i \oplus s$.
   (c) Choose $r_i \xleftarrow{\$} \{0,1\}^*$, compute $\widehat{f}^{\mathrm{off}}_{w_0^i, s_0^i}(r_i)$, $\sigma_i := \mathsf{Com}(r_i)$, $\sigma_i' := \mathsf{Com}(w_1^i)$ and $\sigma_i^* := \mathsf{Com}(s_1^i)$.

   Send $(\widehat{f}^{\mathrm{off}}_{w_0^i, s_0^i}(r_i), \sigma_i, \sigma_i', \sigma_i^*)$ for each $i \in [\lambda]$ along with the second message π_2 of $\mathsf{two\text{-}com}$.
3. $\mathcal{V}$ sends the strings $\widehat{s}_0$ and $\widehat{s}_1$, the third message π_3 of $\mathsf{two\text{-}com}$ to $\mathcal{P}$, together with $\mathsf{ch} \xleftarrow{\$} \{0,1\}^\lambda$ and $\alpha \xleftarrow{\$} \mathbb{F} \setminus \{0\}$.
4. $\mathcal{P}$ sends $a = r + \alpha w$ (in the field $\mathbb{F}$), and does the following for every $i \in [\lambda]$:
   (a) If $\mathsf{ch}_i = 0$, send w_0^i, s_0^i and decommit σ_i.
   (b) If $\mathsf{ch}_i = 1$, send $\widehat{f}^{\mathrm{on}}_{w_0^i, s_0^i}(x || \alpha || a || \widehat{s}_0 || \widehat{s}_1, w_1^i, s_1^i; r_i)$ and the decommitment to σ_i' and σ_i^*.

Check Phase:

– For every $i \in [\lambda]$:
  1. If $\mathsf{ch}_i = 0$, check if the received w_0^i, s_0^i are embedded in the circuit $\widehat{f}^{\mathrm{off}}_{w_0^i, s_0^i}$ computed using randomness r_i. Also check that the decommitment information to σ_i is correct.
  2. If $\mathsf{ch}_i = 1$, run the evaluator for the garbled circuit by providing with $\widehat{f}^{\mathrm{on}}_{w_0^i, s_0^i}$ and $\widehat{f}^{\mathrm{off}}_{w_0^i, s_0^i}$ as inputs and accept if the evaluation outputs $(1, x, w_1^i, s_1^i)$ where w_1^i and s_1^i are obtained from the decommitment to σ_i' and σ_i^*.

Fig. 3. Four round Black-box Commit-and-Prove ZKAoK

Proof. We begin by describing the extractor (having oracle access to a PPT prover $\mathcal{P}^*$) that takes as input an accepted transcript $\mathbb{T}$ and outputs a value $w \in \mathbb{F}$ that occurs in majority of the positions and is such that $R(x, w) = 1$.

The extractor rewinds the cheating prover $\mathcal{P}^*$ to the beginning of the third round and gives different uniformly chosen challenge messages $ch \xleftarrow{\$} \{0,1\}^\lambda$. The extractor stops when it obtains for some $i \in [\lambda]$, two decommitments w_0^i, w_i^1 such that $w_0^i \oplus w_1^i = (w \| r)$ and outputs w if $r + \alpha w = a$ from the main thread.

We will now prove that for any PPT prover, the extracted w is such that $R(x, w) = 1$ – in particular, we will show that for any PPT prover, $\Pr[s = \widehat{s}_0 \text{ or } s = \widehat{s}_1] \leq \mathsf{negl}(\lambda)$. Suppose for contradiction there exists a polynomial $\text{poly}(\cdot)$ and $\beta \in \{0,1\}$ such that $\Pr[s = \widehat{s}_\beta] \geq \frac{1}{\text{poly}(\lambda)}$. We will use this to contradict witness indistinguishability of $\mathsf{two\text{-}com}$, or the hiding of the commitments in $\mathsf{two} - \mathsf{com}$. We consider the following sequence of hybrid experiments.

Hyb_0 corresponds to the real experiment, where the challenger generates the verifier messages according to the honest verifier strategy, such that the commitment $\widehat{s}_0$ is equivocable and the commitment $\widehat{s}_1$ is binding for $\gamma \in \{0,1\}$. It then uses the extraction strategy described above to extract s such that $\Pr[s = \widehat{s}_\beta] \geq \frac{1}{\text{poly}(\lambda)}$ for some $\beta \in \{0,1\}$.

In Hyb_{1a}, the challenger sends messages exactly the same way as Hyb_0, except that it samples $\widehat{s} \xleftarrow{\$} \{0,1\}^\lambda$, and in the third message, equivocates $\widehat{s}_0$ to $\widehat{s}$. The value extracted by the challenger must remain indistinguishable between Hyb_0 and Hyb_1, because of the equivocation property of the commitment to $\widehat{s}_0$. However, since $\widehat{s}$ was chosen uniformly at random and independent of $\widehat{s}_0$, the probability that $\widehat{s}$ equals $s_1^i \oplus s_0^i$ (which are both fixed before $\widehat{s}$ is chosen), is at most $2^{-\lambda}$. Otherwise, $\beta = 1$ and we consider the following sequence of hybrids.

In Hyb_{1b}, the challenger sends messages the same way as Hyb_0, except that the commitment $\widehat{s}_0$ is binding and the commitment $\widehat{s}_1$ is equivocable for $\gamma \in \{0,1\}$. It then uses the extraction strategy described above to extract s. By witness indistinguishability of the argument, this is such that $\Pr[s = \widehat{s}_1] \geq \frac{1}{\text{poly}(\lambda)}$.

In Hyb_2, the challenger sends messages exactly the same way as Hyb_{1b}, except that it samples $\widehat{s} \xleftarrow{\$} \{0,1\}^\lambda$, and in the third message, equivocates $\widehat{s}_1$ to $\widehat{s}$. The value extracted by the challenger must remain indistinguishable between Hyb_{1b} and Hyb_2, because of the equivocation property of the commitment to $\widehat{s}_1$. However, since $\widehat{s}$ was chosen uniformly at random and independent of $\widehat{s}_0$, the probability that $\widehat{s}$ equals $s_1^i \oplus s_0^i$ (which are both committed by the prover even before $\widehat{s}$ is chosen), is at most $2^{-\lambda}$.

We now argue that the extracted w occurs in the majority of positions. Let $\mathbb{T}$ be the accepted protocol transcript. We note that with overwhelming probability over the choice of random challenge $ch \in \mathbb{T}$, at least $\lambda - O(\log^2 \lambda)$ positions are such that $f_{w_0^i, s_0^i}(x, \alpha, w_1^i, s_1^i, a, \widehat{s}_0, \widehat{s}_1) = (1, x, \alpha, w_1^i, s_1^i, (a, \widetilde{a}))$. This follows from the robustness property of the randomized encoding scheme. Let S be the set of positions such that the above is true. Additionally, we showed above that for any i (in particular, for any $i \in S$) $s_0^i \oplus s_1^i$ is not equal to $\widehat{s}_0$ or $\widehat{s}_1$ with overwhelming probability. Thus, we have for each $i \in S$, let $w_1^i \oplus w_0^i = (w_i \| r_i)$ where $R(x, w_i) = 1$. Since $f_{w_0^i, s_0^i}(x, \alpha, w_1^i, s_1^i, a, \widehat{s}_0, \widehat{s}_1) = (1, x, \alpha, w_1^i, s_1^i, (a, \widetilde{a}))$ for every $i \in S$, we now have from the definition of f that, $w_i\alpha + r_i = a$. With overwhelming probability over α, this is possible only if there exists $(w, r) \in \mathbb{F}$ such that $w_i = w$ and $r_i = r$ for all $i \in S$ (by Schwartz-Zippel lemma).

We finally argue that the extractor runs in expected polynomial time. Let p be the probability that conditioned on fixing the first two messages of the main thread the prover $\mathcal{P}^*$ gives an accepted proof. Since the running time of the extractor in each rewind is bounded by some polynomial $\mathrm{poly}(\lambda)$, we have that the expected running time of the extractor is $p.\frac{\mathrm{poly}(\lambda)}{p} = \mathrm{poly}(\lambda)$.

This completes the proof of soundness, and of the argument of knowledge property.

Lemma 5. *The protocol in Figure 3 is zero-knowledge against all PPT verifiers $\mathcal{V}$.*

Proof. We begin with a brief overview of the simulation strategy (for simplicity in this overview we only consider non-aborting verifiers). The simulator runs the verifier on randomly chosen prover message for the second round, and observes the openings $\widehat{s}_0^{(1)}$ and $\widehat{s}_1^{(1)}$. On learning $\widehat{s}_0^{(1)}, \widehat{s}_1^{(1)}$, the simulator rewinds and sends a prover message by setting $s = \widehat{s}_0^{(1)}$. If the verifier responds with $\widehat{s}_0^{(2)} \neq \widehat{s}_0^{(1)}$, the simulator rewinds again and sets $s = \widehat{s}_1^{(1)}$. Denote the response of the verifier in the second rewinding by $\widehat{s}_0^{(3)}, \widehat{s}_1^{(3)}$.

Since the first message of two-com is binding to at least one string, if $\widehat{s}_0^{(2)} \neq \widehat{s}_0^{(1)}$, then with overwhelming probability, it must be the case that the commitment to $\widehat{s}_1$ is binding. In other words, $s = \widehat{s}_1^{(1)} = \widehat{s}_1^{(3)}$ with overwhelming probability. In this case, the simulator uses s as witness to complete the proof. The general simulation strategy is detailed in Figure 4.

Proof of Simulation Security. The proof that the simulated distribution is indistinguishable from the real distribution will rely on the witness indistinguishability of a three round sub-protocol that is being executed within the main protocol. Let us give the details.

It was shown in [21] that the single execution (i.e., for each $i \in [\lambda]$) of the sub-protocol is zero-knowledge with soundness error 1/2. Hence, this sub-protocol is also witness indistinguishable. The parallel repetition of any witness indistinguishable protocol preserves the WI property. This also directly proves that conditioned on not aborting, a real transcript is indistinguishable from an ideal transcript.

Next, we prove that the probability of abort is at most $\mathsf{negl}(\lambda)$-far between the real and ideal worlds. Note that by binding property of two-com, the simulator obtains one opening out of $\widehat{s}_0$ and $\widehat{s}_1$ correctly. Therefore, the simulation proceeds to Step 4(b) after at most two non-aborting rewinds. Now, the simulator rewinds

Input: The simulator Simu and verifier $\mathcal{V}$ have common input x and relation R.

Definition of f: The function f has hardwired a share of the witness w_0, and a share of trapdoor information s_0. It takes as input the instance x, a challenge α, a share of the witness w_1, a share of trapdoor information s_1, a value a, $\widehat{s}_0, \widehat{s}_1$ (recovered from the third message of the $\mathsf{two\text{-}com}$) and does the following:

1. Compute $s = s_0 \oplus s_1$. If $\widehat{s}_0 = s$ or $\widehat{s}_1 = s$, output $(1, x, w_1, s_1)$. Else, continue.
2. Compute $w \| r = w_0 \oplus w_1$. If $a \neq (r + \alpha w)$, output $\perp$. Else, output $(R(x, w), x, w_1, s_1)$.

Protocol:

1. Obtain the first message π_1 of $\mathsf{two\text{-}com}$ from $\mathcal{V}$.
2. Simu chooses $r \xleftarrow{\$} \mathbb{F}$, and $s \leftarrow \{0,1\}^\lambda$. For $i \in [\lambda]$ it does the following:
   (a) Choose w_0^i uniformly at random and compute $w_1^i := w_0^i \oplus 0^{|w|+|r|}$.
   (b) Choose s_0^i uniformly at random and compute $s_1^i := s_0^i \oplus s$.
   (c) Choose $r_i \xleftarrow{\$} \{0,1\}^*$, compute $\widehat{f}^{\text{off}}_{w_0^i, s_0^i}(r_i)$, $\sigma_i := \mathsf{Com}(r_i)$, $\sigma'_i := \mathsf{Com}(w_1^i)$ and $\sigma^*_i := \mathsf{Com}(s_1^i)$.

   Send $(\widehat{f}^{\text{off}}_{w_0^i, s_0^i}(r_i), \sigma_i, \sigma'_i, \sigma^*_i)$ for each $i \in [\lambda]$ and the message π_2 of $\mathsf{two\text{-}com}$.
3. If the verifier $\mathcal{V}$ aborts or does not send a valid message, then abort and end the simulation. Otherwise, obtain strings $\widehat{s}_0$ and $\widehat{s}_1$, the third message π_3 of $\mathsf{two\text{-}com}$ from $\mathcal{V}$, together with $\mathsf{ch} \xleftarrow{\$} \{0,1\}^\lambda$ and $\alpha \xleftarrow{\$} \mathbb{F}$. Set $s = \widehat{s}_0$, and rewind the verifier to the end of Step 1.
4. (a) Repeat the following until the verifier sends a valid message for Step 3.
   – With s set to $\widehat{s}_0$ as described above, compute and send $(\widehat{f}^{\text{off}}_{w_0^i, s_0^i}(r_i), \sigma_i, \sigma'_i, \sigma^*_i)$ for each $i \in [\lambda]$ along with the second message π_2 of $\mathsf{two\text{-}com}$.

   On obtaining a valid message from the verifier, parse it as $\widehat{s}_0^{(2)}, \widehat{s}_1^{(2)}$. If $s = \widehat{s}_0^{(2)}$, continue to Step 5, using s as witness. Else, if $\widehat{s}_1 \neq \widehat{s}_1^{(2)}$, abort and end the simulation. Else, set $s = \widehat{s}_1$ and go to Step 4b.

   (b) Repeat the following until the verifier sends a valid message for Step 3.
   – With s set to $\widehat{s}_1$ as described above, compute and send $(\widehat{f}^{\text{off}}_{w_0^i, s_0^i}(r_i), \sigma_i, \sigma'_i, \sigma^*_i)$ for each $i \in [\lambda]$ along with the second message π_2 of $\mathsf{two\text{-}com}$.

   On obtaining a valid message from the verifier, parse it as $\widehat{s}_0^{(3)}, \widehat{s}_1^{(3)}$. If $s = \widehat{s}_1^{(3)}$, continue to Step 5, using s as witness. Else, abort.
5. Simu sends $a = r + \alpha w$ (in the field $\mathbb{F}$), and for every $i \in [\lambda]$:
   (a) If $\mathsf{ch}_i = 0$, send w_0^i, s_0^i and decommit σ_i.
   (b) If $\mathsf{ch}_i = 1$, send $\widehat{f}^{\text{on}}_{w_0^i, s_0^i}(x||\alpha||a||\widehat{s}_0||\widehat{s}_1, w_1^i, s_1^i; r_i)$ and the decommitment to σ'_i and σ^*_i.

Fig. 4. Simulation Strategy for Black-Box Commit-and-Prove ZKAoK

the verifier in Step $4(b)$: by computational hiding of the second message of the protocol, the probability that the simulated view aborts in these rewindings at the end of Step 3 is at most $p \pm \mathsf{negl}(\lambda)$, where p is the probability that the verifier aborts after the second message, when interacting with an honest prover. Conditioned on the verifier not aborting in step 3, the simulator persists after rewinding until the verifier sends a non-aborting message. Thus, the probability of abort in the ideal world remains $p \pm \mathsf{negl}(\lambda)$.

However, this strategy still suffers from the problem that the simulator may not be expected polynomial time. This issue, akin to [11] is resolved by ensuring that the simulator does not run for too long. Specifically, if the adversary did not abort in Step 3, and the simulator proceeds to the rewinding phase, then it first estimates the value of p, which is the probability that the verifier V^* did not abort given commitments to 0 values. This is done by repeating Steps 2 and 3 of the simulation (with fresh random commitments to all zeroes) until $m = 12\lambda$ successful decommits occur (to the same string q that it decommitted to in the main thread). Then, an estimate $\widetilde{\epsilon}$ of p is taken to be m/T, where T is the overall number of attempts until m successful decommits occured. This suffices to ensure that the probability that $\widetilde{\epsilon}$ is not within a constant factor of p is at most $2^{-\lambda}$. An exact analyses of the probabilities can be found in Section 6.5.3 in [20].

Finally, we can switch the simulator to using the trapdoor witness in the three-round WI sub-protocol. More formally, we consider an intermediate hybrid Hyb_1, where the simulator follows the strategy above but continues to use the real witness in the WI argument.

Claim. Hyb_1 is computationally indistinguishable from the ideal world.

Proof. To prove that Hyb_1 and the ideal world are computationally indistinguishable, we build the following reduction R to the witness indistinguishability of the underlying protocol. The reduction R first completes the experiment by rewinding the adversary using the [11] strategy described above to extract the value $s = \widehat{s}_0$ or $\widehat{s}_1$. Next, the reduction rewinds back to the beginning of Step 2, and commits to s. It obtains the first message of the (delayed-input) WI argument externally, giving it both witnesses w, s. In the third round, it computes $a = r\alpha + w$ externally and obtains the WI argument externally proving at either $a = r\alpha + w$ or $s = \widehat{s}_0$ or $s = \widehat{s}_1$.

Note that if w is used as witness, this corresponds to Hyb_1. If s is used as witness, the only difference between this and the simulation strategy is that the simulator computes a completely at random, instead of computing it as $a = r\alpha + w$. These are perfectly indistinguishable because r completely hides w. Thus, any adversary that distinguishes Hyb_1 from the ideal world breaks the witness indistinguishability of $\mathsf{two\text{-}com}$.

6 Extractable and Equivocal Commitments

We describe direct applications of our black-box commit-and-prove protocols to four round black-box extractable and equivocal commitments.

Extractable Commitments. The construction of black-box commit-and-prove ZK, given in Figure 3, is already an extractable commitment to the witness w (proved in Lemma 4). The hiding of the extractable commitment scheme follows from the computational hiding of the commitment and the zero-knowledge property of the protocol proven in Lemma 5.

Equivocal Commitments. We give a construction of equivocal *bit* commitments in Figure 5. This can be extended to an equivocal string commitments by committing to every bit using the protocol in Figure 5. Our construction of equivocal bit commitment is standard: commit to two bits m_0 and m_1, and prove in zero-knowledge (via our round-optimal black-box commit-and-prove strategy) that $m_0 = m_1$.

Lemma 6. *The protocol in Figure 5 is an equivocal commitment scheme.*

Proof. The (computational) binding property of the scheme follows by the binding property of commit-and-prove ZK. The hiding of the equivocal commitment scheme follows directly based on the computational hiding of Com (since the ZK proofs are not even completed in the commit phase).

The equivocal property is the most interesting, we now describe how this follows by simulating the zero-knowledge proof and generating a commitment to $m_0 \neq m_1$. That is, we consider an intermediate hybrid Hyb_1 where the challenger commits to $m_0 = m_1$ (just as in the real experiment), but starts simulating the underlying proof. This is indistinguishable from the real experiment by simulation security of the commit-and-prove protocol.

In the next hybrid, the challenger continues to simulate the ZK proof but sets $m_0 \neq m_1$. This remains indistinguishable by the computational hiding of com. Note that the challenger can freely equivocate in this experiment. This completes our proof of security.

References

1. Afshar, A., Hu, Z., Mohassel, P., Rosulek, M.: How to efficiently evaluate RAM programs with malicious security. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 702–729. Springer, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015)
2. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. In: 45th FOCS. pp. 166–175. IEEE Computer Society Press, Rome, Italy (Oct 17–19, 2004)
3. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. In: 20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA. pp. 260–274 (2005), `https://doi.org/10.1109/CCC.2005.9`
4. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 12. pp. 784–796. ACM Press, Raleigh, NC, USA (Oct 16–18, 2012)
5. Bellare, M., Jakobsson, M., Yung, M.: Round-Optimal Zero-Knowledge Arguments Based on Any One-Way Function, pp. 280–305. Springer Berlin Heidelberg, Berlin, Heidelberg (1997), `https://doi.org/10.1007/3-540-69053-0_20`

Input: The committer $\mathcal{C}$ has an input $m \in \{0,1\}$.
Definition of f: The function f has hardwired a share of the witness w_0, and a share of trapdoor information s_0. It takes as input a challenge α, a share of the witness w_1, a share of trapdoor information s_1, a value a, $\widehat{s}_0, \widehat{s}_1$ (recovered from the third message of the $\mathsf{two\text{-}com}$) and does the following:

1. Compute $s = s_0 \oplus s_1$. If $\widehat{s}_0 = s$ or $\widehat{s}_1 = s$, output $(1, w_1, s_1, a)$. Else, continue.
2. Compute $m_0 \| m_1 \| r = w_0 \oplus w_1$. If $a \neq (r + \alpha(m_0 \| m_1))$, output $\perp$. Else, output $((m_0 \overset{?}{=} m_1), w_1, s_1, a)$.

Commit Phase:

1. $\mathcal{R}$ picks strings $\widehat{s}_0, \widehat{s}_1 \xleftarrow{\$} \{0,1\}^{2\lambda}$ and sends the first message π_1 of $\mathsf{two\text{-}com}(\widehat{s}_0, \widehat{s}_1)$.
2. $\mathcal{C}$ chooses $r \xleftarrow{\$} \mathbb{F}$, and $s \leftarrow \{0,1\}^\lambda$. For $i \in [\lambda]$ it does the following:
   (a) Choose w_0^i uniformly at random and compute $w_1^i := w_0^i \oplus (m\|m\|r)$.
   (b) Choose s_0^i uniformly at random and compute $s_1^i := s_0^i \oplus s$.
   (c) Choose $r_i \xleftarrow{\$} \{0,1\}^*$, set $\widehat{f}^{\mathrm{off}}_{w_0^i, s_0^i}(r_i)$, $\sigma_i := \mathsf{Com}(r_i)$, $\sigma_i' := \mathsf{Com}(w_1^i)$, $\sigma_i^* := \mathsf{Com}(s_1^i)$.

   Send $(\widehat{f}^{\mathrm{off}}_{w_0^i, s_0^i}(r_i), \sigma_i, \sigma_i', \sigma_i^*)$ for each $i \in [\lambda]$ along with the second message π_2 of $\mathsf{two\text{-}com}$.

Decommit Phase:

1. $\mathcal{R}$ sends the strings $\widehat{s}_0$ and $\widehat{s}_1$, the third message π_3 of $\mathsf{two\text{-}com}$ to $\mathcal{P}$, together with $\mathsf{ch} \xleftarrow{\$} \{0,1\}^\lambda$ and $\alpha \xleftarrow{\$} \mathbb{F} \setminus \{0\}$[8].
2. $\mathcal{C}$ sends $a = r + \alpha(m\|m)$ (in the field $\mathbb{F}$), and does the following for every $i \in [\lambda]$:
   (a) If $\mathsf{ch}_i = 0$, send w_0^i, s_0^i and decommit σ_i.
   (b) If $\mathsf{ch}_i = 1$, send $\widehat{f}^{\mathrm{on}}_{w_0^i, s_0^i}(\alpha || a || \widehat{s}_0 || \widehat{s}_1, w_1^i, s_1^i; r_i)$ and the decommitment to σ_i' and σ_i^*.

Check Phase: The receiver accepts the commitment if the following checks pass:

- For every $i \in [\lambda]$:
  1. If $\mathsf{ch}_i = 0$, check if the received w_0^i, s_0^i are embedded in the circuit $\widehat{f}^{\mathrm{off}}_{w_0^i, s_0^i}$ computed using randomness r_i. Also check that the decommitment information to σ_i is correct.
  2. If $\mathsf{ch}_i = 1$, run the evaluator for the garbled circuit by providing with $\widehat{f}^{\mathrm{on}}_{w_0^i, s_0^i}$ and $\widehat{f}^{\mathrm{off}}_{w_0^i, s_0^i}$ as inputs and accept if the evaluation outputs $(1, w_1^i, s_1^i, a)$ where w_1^i and s_1^i are obtained from the decommitment to σ_i' and σ_i^*.

Fig. 5. Black Box Equivocal Commitment

6. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th ACM STOC. pp. 494–503. ACM Press, Montréal, Québec, Canada (May 19–21, 2002)
7. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Simple, Black-Box Constructions of Adaptively Secure Protocols, pp. 387–402. Springer Berlin Heidelberg, Berlin, Heidelberg (2009), `https://doi.org/10.1007/978-3-642-00457-5_23`
8. Feige, U., Lapidot, D., Shamir, A.: Multiple noninteractive zero knowledge proofs under general assumptions. SIAM J. Comput. 29(1), 1–28 (1999), `https://doi.org/10.1137/S0097539792230010`
9. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: 22nd ACM STOC. pp. 416–426. ACM Press, Baltimore, MD, USA (May 14–16, 1990)
10. Garg, S., Mukherjee, P., Pandey, O., Polychroniadou, A.: The exact round complexity of secure computation. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 448–476. Springer, Heidelberg, Germany, Vienna, Austria (May 8–12, 2016)
11. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. J. Cryptology 9(3), 167–190 (1996)
12. Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. SIAM Journal on Computing 25, 169–192 (1990)
13. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC. pp. 218–229. ACM Press, New York City, NY, USA (May 25–27, 1987)
14. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: 17th ACM STOC. pp. 291–304. ACM Press, Providence, RI, USA (May 6–8, 1985)
15. Goyal, V.: Constant round non-malleable protocols using one way functions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC. pp. 695–704. ACM Press, San Jose, CA, USA (Jun 6–8, 2011)
16. Goyal, V., Lee, C., Ostrovsky, R., Visconti, I.: Constructing non-malleable commitments: A black-box approach. In: 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012. pp. 51–60. IEEE Computer Society (2012), `https://doi.org/10.1109/FOCS.2012.47`
17. Goyal, V., Ostrovsky, R., Scafuro, A., Visconti, I.: Black-box non-black-box zero knowledge. In: Shmoys, D.B. (ed.) 46th ACM STOC. pp. 515–524. ACM Press, New York, NY, USA (May 31 – Jun 3, 2014)
18. Goyal, V., Pandey, O., Richelson, S.: Textbook non-malleable commitments. In: Wichs, D., Mansour, Y. (eds.) Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016. pp. 1128–1141. ACM (2016), `http://doi.acm.org/10.1145/2897518.2897657`
19. Haitner, I.: Semi-honest to malicious oblivious transfer - the black-box way. In: Canetti, R. (ed.) Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008. Lecture Notes in Computer Science, vol. 4948, pp. 412–426. Springer (2008), `https://doi.org/10.1007/978-3-540-78524-8_23`
20. Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols - Techniques and Constructions. Information Security and Cryptography, Springer (2010), `https://doi.org/10.1007/978-3-642-14303-8`

21. Hazay, C., Venkitasubramaniam, M.: On the power of secure two-party computation. In: Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II. pp. 397–429 (2016), http://dx.doi.org/10.1007/978-3-662-53008-5_14
22. Huang, Y., Katz, J., Evans, D.: Efficient secure two-party computation using symmetric cut-and-choose. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 18–35. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013)
23. Huang, Y., Katz, J., Kolesnikov, V., Kumaresan, R., Malozemoff, A.J.: Amortizing garbled circuits. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 458–475. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2014)
24. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: 41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA. pp. 294–304. IEEE Computer Society (2000), https://doi.org/10.1109/SFCS.2000.892118
25. Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions for secure computation. In: Kleinberg, J.M. (ed.) Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006. pp. 99–108. ACM (2006), http://doi.acm.org/10.1145/1132516.1132531
26. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., Sahai, A.: Efficient non-interactive secure computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 406–425. Springer, Heidelberg, Germany, Tallinn, Estonia (May 15–19, 2011)
27. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Johnson, D.S., Feige, U. (eds.) Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007. pp. 21–30. ACM (2007), http://doi.acm.org/10.1145/1250790.1250794
28. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2008)
29. Katz, J., Ostrovsky, R.: Round-optimal secure two-party computation. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 2004)
30. Lindell, Y.: Fast cut-and-choose based protocols for malicious and covert adversaries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 1–17. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013)
31. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg, Germany, Barcelona, Spain (May 20–24, 2007)
32. Lindell, Y., Pinkas, B.: Secure two-party computation via cut-and-choose oblivious transfer. Journal of Cryptology 25(4), 680–722 (Oct 2012)
33. Nielsen, J.B., Orlandi, C.: LEGO for two-party secure computation. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 368–386. Springer, Heidelberg, Germany (Mar 15–17, 2009)

34. Ostrovsky, R., Richelson, S., Scafuro, A.: Round-optimal black-box two-party computation. In: Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II. pp. 339–358 (2015), http://dx.doi.org/10.1007/978-3-662-48000-7_17
35. Ostrovsky, R., Scafuro, A., Venkitasubramaniam, M.: Resettably sound zero-knowledge arguments from OWFs - the (semi) black-box way. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 345–374. Springer, Heidelberg, Germany, Warsaw, Poland (Mar 23–25, 2015)
36. Pass, R., Wee, H.: Black-Box Constructions of Two-Party Protocols from One-Way Functions, pp. 403–418. Springer Berlin Heidelberg, Berlin, Heidelberg (2009), https://doi.org/10.1007/978-3-642-00457-5_24
37. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: 43rd FOCS. pp. 366–375. IEEE Computer Society Press, Vancouver, British Columbia, Canada (Nov 16–19, 2002)
38. Rosen, A.: A note on constant-round zero-knowledge proofs for NP. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 191–202. Springer, Heidelberg, Germany, Cambridge, MA, USA (Feb 19–21, 2004)
39. Rosulek, M.: Must you know the code of f to securely compute f? In: Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings. pp. 87–104 (2012), https://doi.org/10.1007/978-3-642-32009-5_7
40. shelat, a., Shen, C.H.: Two-output secure computation with malicious adversaries. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 386–405. Springer, Heidelberg, Germany, Tallinn, Estonia (May 15–19, 2011)
41. Woodruff, D.P.: Revisiting the efficiency of malicious two-party computation. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 79–96. Springer, Heidelberg, Germany, Barcelona, Spain (May 20–24, 2007)
42. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press, Toronto, Ontario, Canada (Oct 27–29, 1986)

A Robust Randomized Encodings

We reiterate (a simplified variant of) the construction of online-offline robust randomized encodings from [21]. While they describe a complex protocol that uses adaptive garbled circuits in order to provide improved efficiency, in this paper we present a simplication of the scheme that does not rely on adaptive garbling.

The randomized encoding of function f consists of two functions $\widehat{f}^{\text{off}}, \widehat{f}^{\text{on}}$ and makes use of two components.

– Let Eqcom denote a non-interactive equivocal commitment scheme for string commitments, for which a commitment transcript can be opened in two modes. In binding mode, the opening must remain statistically binding against any malicious committer. In equivocal mode, a commitment transcript can be equivocated freely by a simulator.
  Such a scheme can be constructed using any non-interactive statistically binding commitment scheme, where the (honest) commitment algorithm requires committing to each bit of the string twice. In an equivocal mode, the

committer is only required to reveal, for every bit in the string, *one* randomly chosen commitment to the bit. While generating an opening in the equivocal mode, a simulator can commit to two different values for every bit of the string and use these to freely equivocate.

– A garbling scheme for circuits, with its algorithms denoted by $\mathsf{Yao.Garble}$ and $\mathsf{Yao.labels}$.

Using this scheme, the construction of online-offline randomized encodings (which follows the ideas in [21]) is as follows:

– $\widehat{f}^{\text{off}}(r)$ generates a commitment to a Yao's garbled circuit for the function f, using scheme Eqcom. The output of this phase is $\mathsf{Eqcom.Commit}(\mathsf{Yao.Garble}(f; r))$.
– $\widehat{f}^{\text{on}}(r)$ consists of the decommitment information (in equivocal mode) for the garbled circuit, that is, this phase outputs $y = \mathsf{Eqcom.EquivOpen}(\widehat{f}^{\text{off}}(r))$ that was generated in the offline phase. Additionally, given an input x, $\widehat{f}^{\text{on}}(r)$ consists of the wire-labels for this input corresponding to Yao's garbled circuit, that is, it also outputs $\mathsf{Yao.labels}(x; r)$.

Recall that robustness requires that, for a correctly computed $\widehat{f}^{\text{off}}(r)$ (that is, when the commitment to Yao's garbled circuit are generated honestly and the $\mathsf{Eqcom.Commit}$ value is correctly generated), there should not exist *any* (maliciously computed) string $\widehat{f}^{\text{on}}$ such that $(\widehat{f}^{\text{off}}, \widehat{f}^{\text{on}})$ generates an output outside the range of f. This is guaranteed by the perfect correctness of Yao's garbling scheme. We refer the reader to [21] for more details and for schemes with improved efficiency.