Universally-Composable Two-Party Computation in Two Rounds

Omer Horvitz[⋆] and Jonathan Katz[⋆⋆]

Dept. of Computer Science, University of Maryland
{horvitz,jkatz}@cs.umd.edu

Abstract. Round complexity is a central measure of efficiency, and characterizing the round complexity of various cryptographic tasks is of both theoretical and practical importance. We show here a universally-composable (UC) protocol (in the common reference string model) for two-party computation of any functionality, where *both* parties receive output, using only two rounds. (This assumes honest parties are allowed to transmit messages simultaneously in any given round; we obtain a three-round protocol when parties are required to alternate messages.) Our results match the obvious lower bounds for the round complexity of secure two-party computation under any reasonable definition of security, regardless of what setup is used. Thus, our results establish that secure two-party computation can be obtained under a commonly-used setup assumption with *maximal* security (i.e., security under general composition) in a *minimal* number of rounds.

To give but one example of the power of our general result, we observe that as an almost immediate corollary we obtain a two-round UC blind signature scheme, matching a result by Fischlin at Crypto 2006 (though, in contrast to Fischlin, we use specific number-theoretic assumptions).

1 Introduction

Round complexity is an important measure of efficiency for cryptographic protocols, and much research has focused on trying to characterize the round complexity of various tasks such as zero knowledge [GK96a, GK96b], Byzantine agreement [PSL80, FL82, FM97, GM98], Verifiable Secret-Sharing [GIKR01, FGG$^+$06], and secure two-party/multi-party computation [Yao86, BMR90, IK00, Lin01, GIKR02, KOS03, KO04]. (Needless to say, this list is not exhaustive.) Here, we focus on the goal of secure two-party computation. Feasibility results in this case are clearly of theoretical importance, both in their own right and because two-party computation may be viewed as the "base case" for secure computation without honest majority. Results in this case are also of potential practical importance since many interesting cryptographic problems (zero

⋆ Research supported by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under agreement #W911NF-06-3-0001.

⋆⋆ Research supported by NSF CAREER award #0447075 and U.S.-Israel Binational Science Foundation grant #2004240. A portion of this work was done at IPAM.

knowledge, commitment, and — as we will see — blind signatures) can be solved by casting them as specific instances of secure two-party computation.

The round complexity of secure two-party computation in the *stand-alone* setting has been studied extensively. Yao [Yao86] gave a constant-round protocol for the case when parties are honest-but-curious. Goldreich, Micali, and Wigderson [GMW87, Gol04] showed how to obtain a protocol tolerating malicious adversaries; however, their protocol does not run in a constant number of rounds. Lindell [Lin01] gave the first constant-round protocol for secure two-party computation in the presence of malicious adversaries. Katz and Ostrovsky [KO04] showed a five-round protocol for malicious adversaries, and proved a lower bound showing that five rounds are necessary (for black-box proofs of security) when no setup is assumed. (Both the upper and lower bound assume parties talk in alternating rounds.) Two-round protocols for secure two-party computation, where only a single player receives output, have been studied in, e.g., [SYY99, CCKM00]; in particular, Cachin et al. [CCKM00] show a two-round protocol for computing arbitrary functionalities in this case assuming a common reference string (CRS) available to all participating parties.

It is by now well known that protocols secure when run in a stand-alone setting may no longer be secure when many copies of the protocol are run concurrently in an arbitrary manner (possibly among different parties), or when run alongside other protocols in a larger network. To address this issue, researchers have proposed models and definitions that would guarantee security in exactly such settings [PW00, Can01]. In this work, we adopt the model of *universal composability* (UC) introduced by Canetti [Can01].

The initial work of Canetti showed broad feasibility results for UC multi-party computation in the presence of a strict majority of honest players. Unfortunately, subsequent work of Canetti and Fischlin [CF01] showed that even for the case of two parties, one of whom may be malicious, there exist functionalities that cannot be securely computed within the UC framework. Further characterization of all such "impossible-to-realize" two-party functionalities is given by [CKL06]. These impossibility results hold for the "plain" model; in contrast, it is known that these negative results can be bypassed if one is willing to assume some sort of "trusted setup". Various forms of trusted setup have been explored [CF01, BCNP04, HMU05, CDPW07, Katz07], the most common of which is the availability of a CRS to all parties in the network. Under this assumption, universally composable multi-party computation of any (well-formed) functionality is possible for any number of corrupted parties [CLOS02].

The round complexity of UC two-party computation has not been explored in detail. The two-party protocol given in [CLOS02] does not run in a constant number of rounds, though this may be due at least in part to the fact that the goal of their work was security under *adaptive* corruptions (where corruptions may happen at any point during the execution of the protocol, and not necessarily at its outset, as is the case with *passive* corruptions). Indeed, it is a long-standing open question to construct a constant-round protocol for adaptively-secure two-party computation even in the stand-alone setting. Jarecki and Shmatikov [JS07] recently showed a four-round protocol, assuming a CRS, for functionalities that generate output for only one of the parties; they also show a two-round protocol in the random oracle model. Using a standard transformation [Gol04], their protocols can be used to compute two-output functionalities at the cost of an additional round.

Our Results. We show a protocol for securely realizing any (well-formed) two-party functionality in the UC framework using only *two* rounds of communication; we stress that *both parties* may receive output. In our work, we allow both parties to simultaneously send a message in any given round (i.e., when both parties are honest), but prove security against a *rushing* adversary who may observe the other party's message in a given round before sending his own. Although this communication model is non-standard in the two-party setting, it matches the convention used in the study of multi-party protocols and allows for a more accurate characterization of the round complexity. Our result holds under any one of various standard number-theoretic assumptions, and does not rely on random oracles. We assume a CRS but, as we have seen, some form of setup is necessary for two-party computation to be possible. We consider static corruptions only; again, recall that even in the stand-alone setting it is not known how to achieve adaptive security in constant rounds.

We achieve our result via the following steps:

– We first show a two-round protocol (where only one party speaks in each round) for secure computation of any single-output functionality. This protocol is similar to that of Cachin et al. [CCKM00], though our protocol is secure in the UC framework. The protocol relies on Yao's "garbled circuit" technique [Yao86], the two-round oblivious transfer protocol of Tauman [Tau05], and the non-interactive zero-knowledge proofs of De Santis et al. [DDO$^+$01]. Using standard techniques [Gol04, Propositions 7.2.11 and 7.4.4], this immediately implies a three-round protocol (where only one party speaks in each round) for any two-output functionality.
– As our main result, we show how two instances of our initial protocol can be run "in parallel" so as to obtain a two-round protocol (where now both parties speak[1] in each round) *even if both parties are to receive output.* The challenging aspect here is to "bind" the two executions so that each party uses the same input in each of the two protocol instances.

It is not hard to see that one-round secure computation, even if both parties are allowed to speak simultaneously, is impossible under any reasonable definition of security and regardless of any global setup assumption; a similar observation holds for two-round protocols when parties speak in alternate rounds. (It may be possible, however, to obtain such protocols given some preprocessing phase run by the two parties.) Thus, interestingly, the round complexity of our protocols is optimal *for any setting of secure computation* and not "just" for the setting of universal composability with a CRS.

[1] We stress again that our security analysis takes into account a *rushing* adversary.

The low round complexity of our protocol implies round-efficient solutions for various cryptographic tasks. To give an example, we show that blind signatures [Cha82] can be reduced to secure computation of a particular functionality (here, we simplify the prior result of [JL97] to the same effect); thus, as almost an immediate corollary of our result we obtain a two-round blind signature protocol, matching a recent result by Fischlin [Fis06]. Our result has certain technical advantages as compared to Fischlin's work: our scheme can be applied to *any* underlying signature scheme and achieves strong unforgeability "for free" (as long as the underlying signature scheme does); in contrast, Fischlin's result applies to a specific signature scheme and achieves strong unforgeability only with significant additional complications. On the other hand, Fishlin's result holds under more general assumptions.

As a second example, we observe that the evaluation of a trust policy, held by a server, on a set of credentials, held by a client, can be cast as an instance of two-party computation. Applying our protocol yields a solution that provides input privacy to both the client and the server in a minimal number of rounds while preserving security under general composition, a combination of traits not present in current solutions (see [BHS04, LDB03, NT05, LL06, BMC06, FAL06] and references therein). The full version of this work contains a more detailed discussion [Hor07].

2 Framework, Tools, and Assumptions

Preliminaries. Let $X = \{X(k,z)\}_{k \in \mathbb{N}, z \in \{0,1\}^*}$ denote an ensemble of binary distributions, where $X(k,z)$ represents the output of a probabilistic, polynomial time (PPT) algorithm on a *security parameter* k and advice z (the ensemble may be parameterized by additional variables, and the algorithm may take additional inputs). We say that ensembles X, Y are *computationally indistinguishable*, and write $X \overset{c}{\approx} Y$, if for any $a \in \mathbb{N}$ there exists $k_a \in \mathbb{N}$ such that for all $k > k_a$, for all z (and for all values any additional variables parameterizing the ensemble may take), we have $|\Pr[X(k,z) = 1] - \Pr[Y(k,z) = 1]| < k^{-a}$.

Universally Composable Security. We work in the Universal Composability (UC) framework of [Can01]. Our focus is on the two-party, static corruption setting. We highlight a few features of the definition we use that are standard but not universal: (1) The real model offers authenticated communication and universal access to a common reference string. Formally, this corresponds to the $(\mathcal{F}_{\text{AUTH}}, \mathcal{F}_{\text{CRS}})$-hybrid model of [Can01]. (2) Message delivery in both the real and ideal models is carried out by the adversary (contrast with [Can01], where messages between the dummy parties and the ideal functionality in the ideal model are delivered immediately). (3) The ideal functionality is not informed of party corruption by the ideal adversary. We make this choice purely to simplify the exposition; our results extend to the more general setting by the same means employed in [CLOS02] (see section 3.3 there).

Universally Composable Zero Knowledge. We use a standard definition of the *ideal zero-knowledge* functionality $\mathcal{F}_{\mathrm{ZK}}$, following the treatment of [CLOS02]. The functionality, parameterized by a relation R, accepts a statement x to be proven, along with a witness w, from a *prover*; it then forwards x to a *verifier* if and only if $R(x, w) = 1$ (i.e., if and only if it is a correct statement). Looking ahead, our constructions will be presented in the $\mathcal{F}_{\mathrm{ZK}}$-hybrid model.

For the case of static adversaries, De Santis et. al. [DDO$^+$01] give a non-interactive protocol (i.e., consists of a single message from the prover to the verifier) that UC realizes $\mathcal{F}_{\mathrm{ZK}}$ for any NP relation (see also a discussion in [CLOS02, Section 6]); the protocol is given in the CRS model and assumes the existence of enhanced trapdoor-permutations (see [Gol04, Appendix C.1] for a discussion of this assumption).

The Decisional Diffie-Hellman (DDH) Assumption. We use a two-round *oblivious transfer (OT)* protocol as a building block in our constructions; any OT protocol based on *smooth projective hashing for hard subset-membership problems* per Tauman's framework [Tau05] will do. To simplify the exposition, we describe our constructions in terms of a protocol based on the Decisional Diffie-Hellman (DDH) assumption [DH76] which we recall here.

A *group generator* $\mathsf{GroupGen}$ is a PPT which on input $k \in \mathbb{N}$ outputs a description of a cyclic group $\mathcal{G}$ of prime order q, the order q with $|q| \geq k$, and a generator $g \in \mathcal{G}$. Looking ahead, we will want to associate messages of length k with group elements; for simplicity we thus assume that $|q| \geq k$ (alternatively, we could use hashing). We say that the DDH problem *is hard for* $\mathsf{GroupGen}$ if for any PPT algorithm A, the following ensembles are computationally indistinguishable:

$$(1)\quad \left\{(\mathcal{G}, q, g) \overset{R}{\leftarrow} \mathsf{GroupGen}(k); a, b \overset{R}{\leftarrow} \mathbb{Z}_q : A(k, z, \mathcal{G}, q, g, g^a, g^b, g^{ab})\right\}_{k\in\mathbb{N}, z\in\{0,1\}^*}$$

$$(2)\quad \left\{(\mathcal{G}, q, g) \overset{R}{\leftarrow} \mathsf{GroupGen}(k); a, b, c \overset{R}{\leftarrow} \mathbb{Z}_q : A(k, z, \mathcal{G}, q, g, g^a, g^b, g^c)\right\}_{k\in\mathbb{N}, z\in\{0,1\}^*}.$$

Yao's "Garbled Circuit" Technique. Our protocols use the "garbled-circuit" technique of Yao [Yao86, LP04]; we follow [KO04] in abstracting the technique, and refer the reader to [LP04] for a full account. Let F_k be a description of a two-input/single-output circuit whose inputs and output are of length k (the technique easily extends to lengths polynomial in k). Yao's results provide two PPT algorithms:

1. Yao_1 is a randomized algorithm which takes as input a security parameter $k \in \mathbb{N}$, a circuit F_k, and a string $y \in \{0, 1\}^k$. It outputs a *garbled circuit* $\mathsf{Circuit}$ and *input-wire labels* $\{Z_{i,\sigma}\}_{i\in\{1,\ldots,k\},\sigma\in\{0,1\}}$.
2. Yao_2 is a deterministic algorithm which takes as input a security parameter $k \in \mathbb{N}$, a "garbled-circuit" $\mathsf{Circuit}$ and values $\{Z_i\}_{i\in\{1,\ldots,k\}}$ where $Z_i \in \{0, 1\}^k$. It outputs either an invalid symbol $\perp$, or a value $v \in \{0, 1\}^k$.

We informally describe how the above algorithms may be used for secure computation when the participating parties are honest-but-curious. Let P_1 hold input $x = x_1 \dots x_k \in \{0,1\}^k$, P_2 hold input $y \in \{0,1\}^k$, and assume P_1 is to obtain the output $F_k(x,y)$. First, P_2 computes $(\mathsf{Circuit}, \{Z_{i,\sigma}\}_{i,\sigma}) \overset{R}{\leftarrow} \mathsf{Yao}_1(k, F_k, y)$ and sends $\mathsf{Circuit}$ to P_1. Then the players engage in k instances of *Oblivious Transfer*: in the i^{th} instance, P_1 enters with input x_i, P_2 enters with input $(Z_{i,0}, Z_{i,1})$, and P_1 obtains $Z_i \overset{\text{def}}{=} Z_{i,x_i}$ (P_2 learns "nothing" about x_i, and P_1 learns "nothing" about $Z_{i,1-x_i}$). P_1 then computes $v \leftarrow \mathsf{Yao}_2(\mathsf{Circuit}, \{Z_i\}_i)$, and outputs v.

With the above in mind, we describe the properties required of $\mathsf{Yao}_1, \mathsf{Yao}_2$. We first require *correctness*: for any F_k, y, any output $(\mathsf{Circuit}, \{Z_{i,\sigma}\}_{i,\sigma})$ of $\mathsf{Yao}_1(k, F_k, y)$ and any x, we have $F_k(x,y) = \mathsf{Yao}_2(k, \mathsf{Circuit}, \{Z_{i,x_i}\}_i)$. The algorithms also satisfy the following notion of *security*: there exists a PPT *simulator* $\mathsf{Yao\text{-}Sim}$ which takes k, F_k, x, v as inputs, and outputs $\mathsf{Circuit}$ and a set of k input-wire labels $\{Z_i\}_i$; furthermore, for any PPT A, the following two ensembles are computationally indistinguishable:

$$(1)\ \left\{(\mathsf{Circuit}, \{Z_{i,\sigma}\}_{i,\sigma}) \overset{R}{\leftarrow} \mathsf{Yao}_1(k, F_k, y) : A(k, z, x, y, \mathsf{Circuit}, \{Z_{i,x_i}\}_i)\right\}_{\substack{k \in \mathbb{N}, z \in \{0,1\}^* \\ x,y \in \{0,1\}^k}}$$

$$(2)\ \left\{v = F_k(x,y) : A(k, z, x, y, \mathsf{Yao\text{-}Sim}(k, F_k, x, v))\right\}_{\substack{k \in \mathbb{N}, z \in \{0,1\}^* \\ x,y \in \{0,1\}^k}}.$$

3 Round-Efficient UC Two-Party Computation

We begin by describing a two-round (where parties take turns in speaking), UC protocol for computing functionalities that provide output for only one of the parties. The protocol may be compiled into one that UC computes functionalities providing output to both parties at the cost of an additional round, using standard tools. We then show how to bind two instances of the initial protocol so as to obtain a two-round (where both parties may speak at any given round), UC protocol for computing functionalities that provide output to both parties. We conclude by showing that two rounds are necessary.

Our constructions use UC zero-knowledge, Yao's garbled circuit technique, and two-message oblivious transfer (OT) as building blocks. As mentioned earlier, any OT protocol based on *smooth projective hashing for a hard subset-membership problem* per Tauman's framework [Tau05] will do. We stress that such OT protocols satisfy a weaker notion of security than the one needed here; we use zero-knowledge to lift the security guarantees to the level we need. To simplify the exposition, we use a protocol from the framework based on the DDH assumption, simplifying a construction due to Naor and Pinkas [NP01]. We remark that other protocols conforming to Tauman's framework are known to exist under the DDH assumption [AIR01], under the N$^{\text{th}}$-residuosity assumption and under both the Quadratic-Residuosity assumption and the Extended Riemann hypothesis [Tau05].

3.1 A Two-Round Protocol for Single-Output Functionalities

Let $\mathcal{F} = \{F_k\}_{k \in \mathbb{N}}$ be a non-reactive, polynomial-sized, two-party functionality that provides output to a single party, say P_1. To simplify matters, we assume that $\mathcal{F}$ is deterministic; randomized functionalities can be handled using standard tools [Gol04, Prop. 7.4.4]. Without loss of generality, assume that F_k takes two k-bit inputs and produces a k-bit output (the protocol easily extends to input/output lengths polynomial in k). Let $\mathsf{GroupGen}$ be a group generator as in Sect. 2.

Informally, the first round of our protocol is used to set up k instances of oblivious transfer. The second round is used to communicate a "garbled circuit" per Yao's construction, and for completing the oblivious-transfer of circuit input-wire labels that correspond to P_1's input (cf. Sect. 2). To gain more intuition, we sketch a single oblivious transfer instance, assuming both parties are honest (the actual construction accounts for possibly malicious behavior by the parties with the aid of zero-knowledge). Let $\mathcal{G}$ be a group and g a generator, provided by $\mathsf{GroupGen}$. To obtain the label corresponding to an input x_i for wire i, P_1 picks elements a, b uniformly at random from $\mathcal{G}$ and sends P_2 a tuple $(u = g^a, v = g^b, w = g^c)$, where c is set to ab if $x_i = 0$, to $(ab - 1)$ otherwise. Note that if the DDH problem is hard for $\mathsf{GroupGen}$, P_2 will not be able to tell a tuple generated for $x_i = 0$ from one generated for $x_i = 1$, preserving P_1's privacy. Let $Z_{i,\sigma}$ be the label corresponding to input bit σ for wire i. P_2 selects r_0, s_0, r_1, s_1 uniformly at random from $\mathcal{G}$, and sends P_1 two pairs as follows:

$$(K_0 = u^{r_0} \cdot g^{s_0}\ ,\ C_0 = w^{r_0} \cdot v^{s_0} \cdot Z_{i,0})\ ; \text{ and}$$
$$(K_1 = u^{r_1} \cdot g^{s_1}\ ,\ C_1 = (g \cdot w)^{r_1} \cdot v^{s_1} \cdot Z_{i,1}).$$

It is easy to verify that P_1 can obtain Z_{i,x_i} by computing $K_{x_i}^{-b} \cdot C_{x_i}$. Moreover, it can be shown that the tuple (K_{1-x_i}, C_{1-x_i}) is uniformly distributed (over the choice of r_{1-x_i}, s_{1-x_i}), and therefore P_1 "learns nothing" (information-theoretically) about the label corresponding to input $(1-x_i)$ for wire i, preserving P_2's privacy.

In the following, we describe our two-round protocol $\pi_{\mathcal{F}}$ for UC realizing $\mathcal{F}$ in the $\mathcal{F}_{\text{ZK}}$-hybrid model. In our description, we always let i range from 1 to k and σ range from 0 to 1.

Common Reference String: On security parameter $k \in \mathbb{N}$, the CRS is $(\mathcal{G}, q, g) \overset{R}{\leftarrow} \mathsf{GroupGen}(k)$.

First Round: P_1 on inputs $k \in \mathbb{N}$, $x = x_1 \ldots x_k \in \{0,1\}^k$ and sid, proceeds as follows:

1. For every i, chooses a_i, b_i uniformly at random from $\mathbb{Z}_q$, sets:

$$c_i = \begin{cases} a_i b_i & x_i = 0 \\ a_i b_i - 1 & \text{otherwise,} \end{cases}$$

and lets $u_i = g^{a_i}, v_i = g^{b_i}, w_i = g^{c_i}$.

2. P_1 sends

$$(\mathsf{ZK\text{-}prover}, sid \circ 1, (\{u_i, v_i, w_i\}_i, (\mathcal{G}, q, g), k), (x, \{a_i, b_i\}_i))$$

to $\mathcal{F}_{\mathrm{ZK}}^1$, where $\mathcal{F}_{\mathrm{ZK}}^1$ is parameterized by the relation:

$$R_1 = \left\{ ((\{u_i, v_i, w_i\}_i, (\mathcal{G}, q, g), k), (x, \{a_i, b_i\}_i)) \;\middle|\; \begin{array}{l} \forall i, u_i = g^{a_i}, v_i = g^{b_i}, w_i = g^{c_i}, \\ \text{where } c_i = \begin{cases} a_i b_i & x_i = 0 \\ a_i b_i - 1 & \text{otherwise} \end{cases} \end{array} \right\}$$

and is set up such that P_1 is the prover and P_2 is the verifier.

Second Round: P_2, on inputs $k \in \mathbb{N}$, $y = y_1 \dots y_k \in \{0,1\}^k$ and sid, and upon receiving

$$(\mathsf{ZK\text{-}proof}, sid \circ 1, (\{u_i, v_i, w_i\}_i, (\mathcal{G}', q', g'), k'))$$

from $\mathcal{F}_{\mathrm{ZK}}^1$, first verifies that $\mathcal{G}' = \mathcal{G}, q' = q, g' = g$ and $k' = k$. If any of these conditions fail, P_2 ignores the message. Otherwise, it proceeds as follows:

1. Generates a "garbled circuit" (cf. Sect. 2) for F_k, based on its own input y. This involves choosing random coins Ω and computing $(\mathsf{Circuit}, \{Z_{i,\sigma}\}_{i,\sigma}) \leftarrow \mathsf{Yao}_1(k, F_k, y; \Omega)$.
2. For every i and σ, chooses $r_{i,\sigma}, s_{i,\sigma}$ uniformly at random from $\mathbb{Z}_q$, and sets:

$$\begin{array}{l} K_{i,0} = u_i^{r_{i,0}} \cdot g^{s_{i,0}},\, C_{i,0} = w_i^{r_{i,0}} \cdot v_i^{s_{i,0}} \cdot Z_{i,0}; \\ K_{i,1} = u_i^{r_{i,1}} \cdot g^{s_{i,1}},\, C_{i,1} = (g \cdot w_i)^{r_{i,1}} \cdot v_i^{s_{i,1}} \cdot Z_{i,1}. \end{array}$$

3. Sends

$$\left(\mathsf{ZK\text{-}prover}, sid \circ 2, \begin{pmatrix} \mathsf{Circuit}, \{K_{i,\sigma}, C_{i,\sigma}\}_{i,\sigma} \\ (\mathcal{G}, q, g), k, \{u_i, v_i, w_i\}_i \end{pmatrix}, \begin{pmatrix} y, \Omega, \{Z_{i,\sigma}\}_{i,\sigma} \\ \{r_{i,\sigma}, s_{i,\sigma}\}_{i,\sigma} \end{pmatrix}\right)$$

to $\mathcal{F}_{\mathrm{ZK}}^2$, where $\mathcal{F}_{\mathrm{ZK}}^2$ is parameterized by the relation:

$$R_2 = \left\{ \begin{pmatrix} \mathsf{Circuit} & \\ \{K_{i,\sigma}, C_{i,\sigma}\}_{i,\sigma} & y, \Omega \\ (\mathcal{G}, q, g), k & , \{Z_{i,\sigma}\}_{i,\sigma} \\ \{u_i, v_i, w_i\}_i & \{r_{i,\sigma}, s_{i,\sigma}\}_{i,\sigma} \end{pmatrix} \;\middle|\; \begin{array}{c} (\mathsf{Circuit}, \{Z_{i,\sigma}\}_{i,\sigma}) = \mathsf{Yao}_1(k, F_k, y; \Omega) \\ \wedge \forall i, \\ K_{i,0} = u_i^{r_{i,0}} \cdot g^{s_{i,0}}, C_{i,0} = w_i^{r_{i,0}} \cdot v_i^{s_{i,0}} \cdot Z_{i,0}; \\ K_{i,1} = u_i^{r_{i,1}} \cdot g^{s_{i,1}}, C_{i,1} = (g \cdot w_i)^{r_{i,1}} \cdot v_i^{s_{i,1}} \cdot Z_{i,1} \end{array} \right\}$$

and is set up such that P_2 is the prover and P_1 is the verifier.

Output Computation: P_1, upon receipt of message

$$(\mathsf{ZK\text{-}proof}, sid \circ 2, (\mathsf{Circuit}, \{K_{i,\sigma}, C_{i,\sigma}\}_{i,\sigma}, (\mathcal{G}', q', g'), k', \{u_i', v_i', w_i'\}_i))$$

from $\mathcal{F}_{\mathrm{ZK}}^2$, first verifies that $\mathcal{G}' = \mathcal{G}, q' = q, g' = g, k' = k$ and $\{u_i', v_i', w_i'\}_i = \{u_i, v_i, w_i\}_i$. If any of these conditions fail, P_1 ignores the message. Otherwise, it completes the protocol by computing $Z_i \overset{\text{def}}{=} K_{i,x_i}^{-b_i} \cdot C_{i,x_i}$, computing $v \leftarrow \mathsf{Yao}_2(k, \mathsf{Circuit}, \{Z_i\}_i)$ and reporting v as output if $v \neq \perp$.

Concrete round complexity. When composed with the non-interactive protocol of De Santis et al. [DDO+01] UC-realizing F_{ZK}, our protocol takes two communication rounds. Its security now additionally rests on the existence of enhanced trapdoor permutations.

Security. The protocol may be viewed as a degenerate version of the construction we present next, and its security follows in a straightforward manner from security of the latter.

3.2 A Two-Round Protocol for Two-Output Functionalities

Let $\mathcal{F} = \left\{ F_k \stackrel{\mathrm{def}}{=} (F_k^1, F_k^2) \right\}_{k \in \mathbb{N}}$ be a non-reactive, polynomial-sized, two-party functionality such that P_1 wishes to obtain $F_k^1(x, y)$ and P_2 wishes to obtain $F_k^2(x, y)$ when P_1 holds x and P_2 holds y. Without loss of generality, assume once more that $\mathcal{F}$ is deterministic; that x, y and the outputs of F_k^1, F_k^2 are k-bit strings; and that $\mathsf{GroupGen}$ is as in Sect. 2.

The protocol of the preceding section provides means to securely compute a functionality that provides output to *one* of the parties, in two rounds. To securely-compute our two-output functionality $F_k = (F_k^1, F_k^2)$, we run one instance of that protocol such that P_1 receives F_k^1 (with a first-round message originating from P_1 and a second-round message from P_2), and a second instance such that P_2 receives F_k^2 (with a first-round message originating from P_2 and a second-round message from P_1); if we allow the parties to transmit messages *simultaneously* in any given round, this yields a two-round protocol. All that's left to ensure is that each party enters both instances of the protocol with the same input. Here, we have the relation parameterizing the second round zero-knowledge functionality enforce this condition[2].

Below, we describe our two-round protocol $\pi_{\mathcal{F}}$ for UC realizing $\mathcal{F}$ in the $\mathcal{F}_{\mathrm{ZK}}$-hybrid model when parties are allowed to send messages simultaneously in any given round. We describe our protocol from the perspective of P_1; P_2 behaves analogously (i.e., the protocol is symmetric). In the description, we always let i range from 1 to k and σ range from 0 to 1.

Common Reference String: On security parameter $k \in \mathbb{N}$, the CRS is $(\mathcal{G}, q, g) \stackrel{R}{\leftarrow} \mathsf{GroupGen}(k)$.

First Round: P_1 on inputs $k \in \mathbb{N}$, $x = x_1 \ldots x_k \in \{0,1\}^k$ and *sid*, proceeds as follows:

[2] Alternatively, we can make the following modifications to the initial protocol: each party will add a commitment to its input to its original protocol message, and modify its zero-knowledge assertion to reflect that it has constructed its initial message with an input that is consistent with the commitment. Two instances of this protocol can now be run in parallel as above without further modifications (note that the second-round commitments become redundant). We omit the details here.

1. For every i, chooses a_i, b_i uniformly at random from $\mathbb{Z}_q$, sets:
$$c_i = \begin{cases} a_i b_i & x_i = 0 \\ a_i b_i - 1 & \text{otherwise,} \end{cases}$$
and lets $u_i = g^{a_i}, v_i = g^{b_i}, w_i = g^{c_i}$.
2. Sends
$$(\mathsf{ZK\text{-}prover}, \mathit{sid} \circ 1 \circ P_1, (\{u_i, v_i, w_i\}_i, (\mathcal{G}, q, g), k), (x, \{a_i, b_i\}_i))$$
to $\mathcal{F}_{\mathrm{ZK}}^{1, P_1 \to P_2}$, where $\mathcal{F}_{\mathrm{ZK}}^{1, P_1 \to P_2}$ is parameterized by the relation:
$$R_1 = \left\{ ((\{u_i, v_i, w_i\}_i, (\mathcal{G}, q, g), k), (x, \{a_i, b_i\}_i)) \;\middle|\; \begin{array}{l} \forall i, u_i = g^{a_i}, v_i = g^{b_i}, w_i = g^{c_i}, \\ \text{where } c_i = \begin{cases} a_i b_i & x_i = 0 \\ a_i b_i - 1 & \text{otherwise} \end{cases} \end{array} \right\}$$
and is set up such that P_1 is the prover and P_2 is the verifier.

Second Round: Upon receiving the symmetric first-round message
$$(\mathsf{ZK\text{-}proof}, \mathit{sid} \circ 1 \circ P_2, (\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, (\mathcal{G}', q', g'), k'))$$
from $\mathcal{F}_{\mathrm{ZK}}^{1, P_2 \to P_1}$ (defined analogously to $\mathcal{F}_{\mathrm{ZK}}^{1, P_1 \to P_2}$ using the relation R_1, but set up such that P_2 is the prover and P_1 is the verifier), P_1 verifies that $\mathcal{G}' = \mathcal{G}, q' = q, g' = g$ and $k' = k$. If any of these conditions fail, P_1 ignores the message. Otherwise, it proceeds as follows:

1. Generates a "garbled circuit" (cf. Sect. 2) for F_k^2, based on its own input x. This involves choosing random coins Ω and computing $(\mathsf{Circuit}, \{Z_{i,\sigma}\}_{i,\sigma}) \leftarrow \mathsf{Yao}_1(k, F_k^2, x; \Omega)$.
2. For every i and σ, chooses $r_{i,\sigma}, s_{i,\sigma}$ uniformly at random from $\mathbb{Z}_q$, and sets:
$$\begin{array}{l} K_{i,0} = \bar{u}_i^{r_{i,0}} \cdot g^{s_{i,0}},\ C_{i,0} = \bar{w}_i^{r_{i,0}} \cdot \bar{v}_i^{s_{i,0}} \cdot Z_{i,0}; \\ K_{i,1} = \bar{u}_i^{r_{i,1}} \cdot g^{s_{i,1}},\ C_{i,1} = (g \cdot \bar{w}_i)^{r_{i,1}} \cdot \bar{v}_i^{s_{i,1}} \cdot Z_{i,1}. \end{array}$$
3. Sends
$$\left(\mathsf{ZK\text{-}prover}, \mathit{sid} \circ 2 \circ P_1, \begin{pmatrix} \mathsf{Circuit}, \{K_{i,\sigma}, C_{i,\sigma}\}_{i,\sigma} \\ (\mathcal{G}, q, g), k, \{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i \\ \{u_i, v_i, w_i\}_i \end{pmatrix}, \begin{pmatrix} x, \Omega, \{Z_{i,\sigma}\}_{i,\sigma} \\ \{r_{i,\sigma}, s_{i,\sigma}\}_{i,\sigma} \\ \{a_i, b_i\}_i \end{pmatrix} \right)$$
to $\mathcal{F}_{\mathrm{ZK}}^{2, P_1 \to P_2}$, where $\mathcal{F}_{\mathrm{ZK}}^{2, P_1 \to P_2}$ is parameterized by the relation:
$$R_2 = \left\{ \begin{pmatrix} \begin{array}{c} \mathsf{Circuit} \\ \{K_{i,\sigma}, C_{i,\sigma}\}_{i,\sigma} \\ (\mathcal{G}, q, g), k \\ \{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i \\ \{u_i, v_i, w_i\}_i \end{array}, \begin{array}{c} x, \Omega \\ \{Z_{i,\sigma}\}_{i,\sigma} \\ \{r_{i,\sigma}, s_{i,\sigma}\}_{i,\sigma} \\ \{a_i, b_i\}_i \end{array} \end{pmatrix} \;\middle|\; \begin{array}{c} (\mathsf{Circuit}, \{Z_{i,\sigma}\}_{i,\sigma}) = \mathsf{Yao}_1(k, F_k^2, x; \Omega) \\ \wedge\ \forall i, \\ K_{i,0} = \bar{u}_i^{r_{i,0}} \cdot g^{s_{i,0}},\ C_{i,0} = \bar{w}_i^{r_{i,0}} \cdot \bar{v}_i^{s_{i,0}} \cdot Z_{i,0} \\ K_{i,1} = \bar{u}_i^{r_{i,1}} \cdot g^{s_{i,1}},\ C_{i,1} = (g \cdot \bar{w}_i)^{r_{i,1}} \cdot \bar{v}_i^{s_{i,1}} \cdot Z_{i,1} \\ \wedge\ \forall i,\ u_i = g^{a_i}, v_i = g^{b_i}, w_i = g^{c_i}, \\ \text{where } c_i = \begin{cases} a_i b_i & x_i = 0 \\ a_i b_i - 1 & \text{otherwise} \end{cases} \end{array} \right\}$$
and is set up such that P_1 is the prover and P_2 is the verifier.

Output Computation: Upon receiving the symmetric second-round message

$$(\mathsf{ZK\text{-}proof}, sid{\circ}2{\circ}P_2, (\overline{\mathsf{Circuit}}, \left\{\bar{K}_{i,\sigma}, \bar{C}_{i,\sigma}\right\}_{i,\sigma}, (\mathcal{G}', q', g'), k', \{u'_i, v'_i, w'_i\}_i, \{\bar{u}'_i, \bar{v}'_i, \bar{w}'_i\}_i))$$

from $\mathcal{F}_{\text{ZK}}^{2,P_2\rightarrow P_1}$ (defined analogously to $\mathcal{F}_{\text{ZK}}^{2,P_1\rightarrow P_2}$ using the relation R_2, but set up such that P_2 is the prover and P_1 is the verifier), P_1 verifies that $\mathcal{G}' = \mathcal{G}, q' = q, g' = g, k' = k$, that $\{u'_i, v'_i, w'_i\}_i = \{u_i, v_i, w_i\}_i$ and that $\{\bar{u}'_i, \bar{v}'_i, \bar{w}'_i\}_i = \{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i$. If any of these conditions fail, P_1 ignores the message. Otherwise, it completes the protocol by computing $\bar{Z}_i \overset{\text{def}}{=} \bar{K}_{i,x_i}^{-b_i} \cdot \bar{C}_{i,x_i}$, computing $v \leftarrow \mathsf{Yao}_2(k, \overline{\mathsf{Circuit}}, \left\{\bar{Z}_i\right\}_i)$ and reporting v as output if $v \neq \bot$.

Concrete round complexity. As in our first protocol, this takes two rounds when composed with the protocol of De Santis et al. [DDO$^+$01] realizing $\mathcal{F}_{\text{ZK}}$; the security of our protocols now additionally relies on the existence of enhanced trapdoor permutations.

Theorem 1. *Assuming that the DDH problem is hard for* $\mathsf{GroupGen}$*, the above protocol UC-realizes* $\mathcal{F}$ *in the* $\mathcal{F}_{ZK}$*-hybrid model (in the presence of static adversaries).*

Let $\mathcal{A}$ be a (static) adversary operating against $\pi_{\mathcal{F}}$ in the $\mathcal{F}_{\text{ZK}}$-hybrid model. To prove the theorem, we construct a simulator $\mathcal{S}$ such that no environment $\mathcal{Z}$ can tell with a non-negligible probability whether it is interacting with $\mathcal{A}$ and P_1, P_2 running $\pi_{\mathcal{F}}$ in the $\mathcal{F}_{\text{ZK}}$-hybrid model or with $\mathcal{S}$ and $\tilde{P}_1, \tilde{P}_2$ in the ideal process for $\mathcal{F}$. $\mathcal{S}$ will internally run a copy of $\mathcal{A}$, "simulating" for it an execution of $\pi_{\mathcal{F}}$ in the $\mathcal{F}_{\text{ZK}}$-hybrid model (by simulating an environment, a CRS, ideal $\mathcal{F}_{\text{ZK}}$ functionalities and parties P_1, P_2) that matches $\mathcal{S}$'s view of the ideal process; $\mathcal{S}$ will use $\mathcal{A}$'s actions to guide its own in the ideal process. We refer to an event as occurring in the *internal simulation* if it happens within the execution environment that $\mathcal{S}$ simulates for A. We refer to an event as occurring in the *external process* if it happens within the ideal process, in which $\mathcal{S}$ is participating. $\mathcal{S}$ proceeds as follows:

Initial activation. $\mathcal{S}$ sets the (simulated) CRS to be $(\mathcal{G}, q, g) \overset{R}{\leftarrow} \mathsf{GroupGen}(k)$. It copies the input value written by $\mathcal{Z}$ on its own input tape onto $\mathcal{A}$'s input tape and activates $\mathcal{A}$. If $\mathcal{A}$ corrupts party P_i (in the internal simulation), $\mathcal{S}$ corrupts $\tilde{P}_i$ (in the external process). When $\mathcal{A}$ completes its activation, $\mathcal{S}$ copies the output value written by $\mathcal{A}$ on its output tape to $\mathcal{S}$'s own output tape, and ends its activation.

P_2 **only is corrupted.** Upon activation, $\mathcal{S}$ copies the input value written by $\mathcal{Z}$ on its own input tape onto $\mathcal{A}$'s input tape. In addition, if $\tilde{P}_1$ has added a message $(\mathcal{F}\text{-}\mathsf{input}_1, sid, \cdot)$ for $\mathcal{F}$ to its outgoing communication tape (in the external process; recall that $\mathcal{S}$ can only read the *public headers* of messages on the outgoing communication tapes of uncorrupted dummy parties), $\mathcal{S}$, for every i, chooses a_i, b_i uniformly at random from $\mathbb{Z}_q$, sets $u_i = g^{a_i}, v_i = g^{b_i}, w_i = g^{a_i b_i}$ for future use, and adds a message $(\mathsf{ZK\text{-}prover}, sid \circ 1 \circ P_1, \bot, \bot)$ for $\mathcal{F}_{\text{ZK}}^{1,P_1\rightarrow P_2}$

to P_1's outgoing communication tape (in the internal simulation; recall that A will only be able to read the *public header* of a message intended for $\mathcal{F}_{\text{ZK}}$ on the outgoing communication tape of an uncorrupted party in the $\mathcal{F}_{\text{ZK}}$-hybrid model). $\mathcal{S}$ then activates $\mathcal{A}$.

Upon completion of $\mathcal{A}$'s activation, $\mathcal{S}$ acts as follows:

1. If $\mathcal{A}$ delivered the message $(\mathsf{ZK\text{-}prover}, sid \circ 1 \circ P_1, \perp, \perp)$ from P_1 to $\mathcal{F}_{\text{ZK}}^{1,P_1 \to P_2}$ (in the internal simulation), $\mathcal{S}$ adds the message
$$(\mathsf{ZK\text{-}proof}, sid \circ 1 \circ P_1, (\{u_i, v_i, w_i\}_i, (\mathcal{G}, q, g), k))$$
for P_2 and $\mathcal{A}$ to $\mathcal{F}_{\text{ZK}}^{1,P_1 \to P_2}$'s outgoing communication tape (in the internal simulation). Informally, $\mathcal{S}$ constructs the message from $\mathcal{F}_{\text{ZK}}^{1,P_1 \to P_2}$ to P_2 and $\mathcal{A}$ (in the internal simulation) in accordance with $\pi_{\mathcal{F}}$, except that it always lets w_i be $g^{a_i b_i}$.
2. If $\mathcal{A}$ delivered a message
$$(\mathsf{ZK\text{-}prover}, sid \circ 1 \circ P_2, (\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, (\mathcal{G}', q', g'), k'), (y, \{\bar{a}_i, \bar{b}_i\}_i))$$
from P_2 to $\mathcal{F}_{\text{ZK}}^{1,P_2 \to P_1}$ (in the internal simulation), $\mathcal{S}$ verifies that
$$((\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, (\mathcal{G}', q', g'), k'), (y, \{\bar{a}_i, \bar{b}_i\}_i)) \in R_1.$$
If the verification fails, $\mathcal{S}$ does nothing. Otherwise, $\mathcal{S}$ adds the message
$$(\mathsf{ZK\text{-}proof}, sid \circ 1 \circ P_2, (\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, (\mathcal{G}', q', g'), k'))$$
for P_1 and $\mathcal{A}$ to $\mathcal{F}_{\text{ZK}}^{1,P_2 \to P_1}$'s outgoing communication tape (in the internal simulation), and delivers the message $(\mathcal{F}\text{-}\mathsf{input}_2, sid, y)$ from (the corrupted) $\tilde{P}_2$ to $\mathcal{F}$ (in the external simulation). $\mathcal{S}$ records the values y and $\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i$.
3. If $\mathcal{A}$ delivered the message
$$(\mathsf{ZK\text{-}proof}, sid \circ 1 \circ P_2, (\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, (\mathcal{G}', q', g'), k'))$$
from $\mathcal{F}_{\text{ZK}}^{1,P_2 \to P_1}$ to P_1 (in the internal simulation), $\mathcal{S}$ first verifies that $\tilde{P}_1$ has a message $(\mathcal{F}\text{-}\mathsf{input}_1, sid, \cdot)$ for $\mathcal{F}$ on its outgoing communication tape (in the external process) and that $\mathcal{G}' = \mathcal{G}, q' = q, g' = g$ and $k' = k$. If any of these fail, $\mathcal{S}$ does nothing. Otherwise, it adds the message $(\mathsf{ZK\text{-}prover}, sid \circ 2 \circ P_1, \perp, \perp)$ for $\mathcal{F}_{\text{ZK}}^{2,P_1 \to P_2}$ to P_1's outgoing communication tape (in the internal simulation), delivers $(\mathcal{F}\text{-}\mathsf{input}_1, sid, \cdot)$ from $\tilde{P}_1$ to $\mathcal{F}$ (in the external process), and notes to itself that the Round-1 message from $\mathcal{F}_{\text{ZK}}^{1,P_2 \to P_1}$ to P_1 (in the internal simulation) has been delivered. Note that once the activation of $\mathcal{S}$ will be complete, $\mathcal{F}$ will be in possession of both its inputs and will be activated next (in the external process).
4. If $\mathcal{A}$ delivered the message $(\mathsf{ZK\text{-}prover}, sid \circ 2 \circ P_1, \perp, \perp)$ from P_1 to $\mathcal{F}_{\text{ZK}}^{2,P_1 \to P_2}$, $\mathcal{S}$ proceeds as follows. First note that at this point, we are guaranteed that two inputs were delivered to $\mathcal{F}$ and that $\mathcal{F}$ has been activated subsequently (in the external process); therefore, $\mathcal{F}$ has written a message

$(\mathcal{F}\text{-}\mathsf{output}_2, sid, v)$ for $\tilde{P}_2$ on its outgoing communication tape (in the external process; note that $\mathcal{S}$ may read the contents of a message from $\mathcal{F}$ to a corrupted party). Also note that at this point, $\mathcal{S}$ has recorded values y and $\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i$ sent by (the corrupted) P_2 in its first-round message to $\mathcal{F}_{\mathrm{ZK}}^{1,P_2\to P_1}$. $\mathcal{S}$ produces a simulated "garbled circuit" and input-wire labels using F_k^2, y and v (cf. Sect. 2) by computing $(\mathsf{Circuit}, \{Z_i\}_i) \overset{R}{\leftarrow} \mathsf{Yao\text{-}Sim}(k, F_k^2, y, v)$. For every i, it chooses r_{i,y_i}, s_{i,y_i} uniformly at random from $\mathbb{Z}_q$, sets:

$$K_{i,y_i} = \bar{u}_i^{r_{i,y_i}} \cdot g^{s_{i,y_i}}$$
$$C_{i,y_i} = \begin{cases} \bar{w}_i^{r_{i,y_i}} \cdot \bar{v}_i^{s_{i,y_i}} \cdot Z_i & \text{if } y_i = 0 \\ (g \cdot \bar{w}_i)^{r_{i,y_i}} \cdot \bar{v}_i^{s_{i,y_i}} \cdot Z_i & \text{otherwise,} \end{cases}$$

and sets $K_{i,1-y_i}, C_{i,1-y_i}$ to be elements selected uniformly at random from $\mathcal{G}$. It then adds the message

$$\left(\mathsf{ZK\text{-}proof}, sid \circ 2 \circ P_1, \begin{array}{c} \mathsf{Circuit}, \{K_{i,\sigma}, C_{i,\sigma}\}_{i,\sigma} \\ (\mathcal{G}, q, g), k, \{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i \\ \{u_i, v_i, w_i\}_i \end{array}\right)$$

for P_2 and $\mathcal{A}$ to the outgoing communication tape of $\mathcal{F}_{\mathrm{ZK}}^{2,P_1\to P_2}$. Informally, $\mathcal{S}$ constructs the message in accordance with $\pi_{\mathcal{F}}$, except that it uses simulated circuit and input wire labels, and sets $\{K_{i,1-y_i}, C_{i,1-y_i}\}_i$ to be uniform elements in $\mathcal{G}$.

5. If $\mathcal{A}$ delivered a message

$$\left(\mathsf{ZK\text{-}prover}, sid \circ 2 \circ P_2 \quad, \begin{array}{c} \overline{\mathsf{Circuit}}, \{\bar{K}_{i,\sigma}, \bar{C}_{i,\sigma}\}_{i,\sigma} \\ (\mathcal{G}', q', g'), k', \{u'_i, v'_i, w'_i\}_i \\ \{\bar{u}'_i, \bar{v}'_i, \bar{w}'_i\}_i \end{array}, \begin{array}{c} y', \bar{\Omega}, \{\bar{Z}_{i,\sigma}\}_{i,\sigma} \\ \{\bar{r}_{i,\sigma}, \bar{s}_{i,\sigma}\}_{i,\sigma} \\ \{\bar{a}'_i, \bar{b}'_i\}_i \end{array}\right)$$

from $\mathcal{A}$ to $\mathcal{F}_{\mathrm{ZK}}^{2,P_2\to P_1}$ (in the internal simulation), $\mathcal{S}$ verifies that

$$\left(\begin{array}{c} \overline{\mathsf{Circuit}}, \{\bar{K}_{i,\sigma}, \bar{C}_{i,\sigma}\}_{i,\sigma} \\ (\mathcal{G}', q', g'), k', \{u'_i, v'_i, w'_i\}_i \\ \{\bar{u}'_i, \bar{v}'_i, \bar{w}'_i\}_i \end{array}, \begin{array}{c} y', \bar{\Omega}, \{\bar{Z}_{i,\sigma}\}_{i,\sigma} \\ \{\bar{r}_{i,\sigma}, \bar{s}_{i,\sigma}\}_{i,\sigma} \\ \{\bar{a}'_i, \bar{b}'_i\}_i \end{array}\right) \in R_2.$$

If the verification fails, $\mathcal{S}$ does nothing. Otherwise, $\mathcal{S}$ adds the message

$$\left(\mathsf{ZK\text{-}proof}, sid \circ 2 \circ P_2, \begin{array}{c} \overline{\mathsf{Circuit}}, \{\bar{K}_{i,\sigma}, \bar{C}_{i,\sigma}\}_{i,\sigma} \\ (\mathcal{G}', q', g'), k', \{u'_i, v'_i, w'_i\}_i \\ \{\bar{u}'_i, \bar{v}'_i, \bar{w}'_i\}_i \end{array}\right)$$

for P_1 and $\mathcal{A}$ to $\mathcal{F}_{\mathrm{ZK}}^{2,P_2\to P_1}$'s outgoing communication tape (in the internal simulation).

6. If $\mathcal{A}$ delivered the message

$$\left(\mathsf{ZK\text{-}proof}, sid \circ 2 \circ P_2, \begin{array}{c} \overline{\mathsf{Circuit}}, \{\bar{K}_{i,\sigma}, \bar{C}_{i,\sigma}\}_{i,\sigma} \\ (\mathcal{G}', q', g'), k', \{u'_i, v'_i, w'_i\}_i \\ \{\bar{u}'_i, \bar{v}'_i, \bar{w}'_i\}_i \end{array}\right)$$

from $\mathcal{F}_{\mathrm{ZK}}^{2,P_2\to P_1}$ to P_1 (in the internal simulation), $\mathcal{S}$ first checks whether a Round-1 message from $\mathcal{F}_{\mathrm{ZK}}^{1,P_2\to P_1}$ to P_1 (in the internal simulation) has been delivered, per Item 3 above; if not, $\mathcal{S}$ does nothing. Otherwise, we are guaranteed that two inputs were delivered to $\mathcal{F}$, and that $\mathcal{F}$ has subsequently been activated and written a message $(\mathcal{F}\text{-}\mathsf{output}_1, sid, \cdot)$ for $\tilde{P}_1$ on its outgoing communication tape (in the external process). $\mathcal{S}$ verifies that $\mathcal{G}' = \mathcal{G}, q' = q, g' = g, k' = k$, that $\{u'_i, v'_i, w'_i\}_i = \{u_i, v_i, w_i\}_i$ and that $\{\bar{u}'_i, \bar{v}'_i, \bar{w}'_i\}_i = \{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i$ (intuitively, the checks, along those performed by $\mathcal{S}$ on behalf of $\mathcal{F}_{\mathrm{ZK}}^{2,P_2\to P_1}$ per Item 5 above, guarantee that (the corrupted) P_2 has used the same input consistently in both rounds, i.e., that $y' = y$); if so, $\mathcal{S}$ delivers the message $(\mathcal{F}\text{-}\mathsf{output}_1, sid, \cdot)$ from $\mathcal{F}$ to $\tilde{P}_1$ (in the external process).

After performing one of the above (if any), $\mathcal{S}$ copies the output value written by $\mathcal{A}$ on its output tape to $\mathcal{S}$'s own output tape, and ends its activation.

Other corruption scenarios. $\mathcal{S}$'s actions for the case where P_1 is corrupted are symmetric to the above; its actions for the case where both parties are corrupted and for the case where neither is, are straightforward.

This concludes the description of $\mathcal{S}$. We claim that for any $\mathcal{Z}$:

$$\mathrm{EXEC}^{\mathcal{F}_{\mathrm{ZK}}}_{\pi_{\mathcal{F}},\mathcal{A},\mathcal{Z}} \overset{\mathrm{c}}{\approx} \mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}, \tag{1}$$

We prove the above in cases corresponding to the parties $\mathcal{A}$ corrupts. We give here an informal description of the case where P_2 only is corrupted (the case where P_1 only is corrupted is symmetric, and the cases where either or neither parties are corrupted are straightforward); refer to [Hor07] for the complete proof.

Loosely speaking, when P_2 only is corrupted, the following differences between a real-life execution of $\pi_{\mathcal{F}}$ among P_1, P_2 in the $\mathcal{F}_{\mathrm{ZK}}$-hybrid model and the ideal process for $\mathcal{F}$ among $\tilde{P}_1, \tilde{P}_2$ may be noted: (1) in the former, P_1 computes its output based on a "garbled circuit" and obliviously-transferred input-wire labels corresponding to its input, received in the second round of the protocol, while in the latter, $\tilde{P}_1$ receives its output from $\mathcal{F}$ based on the value y that $\mathcal{S}$ obtained while simulating $\mathcal{F}_{\mathrm{ZK}}^{1,P_2\to P_1}$ for the first round of the protocol; (2) in the former, the first round message from $F_{\mathrm{ZK}}^{1,P_1\to P_2}$ to P_2 contains values $w_i = g^{c_i}$ where $c_i = a_i b_i$ when $x_i = 0$, $c_i = a_i b_i - 1$ when $x_i = 1$, while in the latter, the message (in the internal simulation) contains $w_i = g^{a_i b_i}$ for all i; (3) in the former, the second-round message from $\mathcal{F}_{\mathrm{ZK}}^{2,P_1\to P_2}$ to P_2 contains values $K_{i,(1-y_i)}, C_{i,(1-y_i)}$ computed as in the specification of the protocol, while in the latter, those values (in the internal simulation) are chosen uniformly at random from $\mathcal{G}$; and (4) in the former, Yao_1 is used to compute the "garbled circuit" and input-wire labels for the second-round message from $\mathcal{F}_{\mathrm{ZK}}^{2,P_1\to P_2}$ to P_2, while in the latter, $\mathsf{Yao\text{-}Sim}$ is used for that purpose, based on P_2's output from $\mathcal{F}(x, y)$, where y was obtained by $\mathcal{S}$ while simulating $F_{\mathrm{ZK}}^{1,P_2\to P_1}$ for the first round of the protocol.

Nevertheless, we claim that Eq. 1 holds, based on (1) the correctness of Yao's "garbled circuit" technique, the correctness of the oblivious transfer protocol and the enforcement of parties entering the two rounds of the protocol with a consistent input; (2) the hardness of the DDH assumption for $\mathsf{GroupGen}$; (3) the uniformity of $K_{i,(1-y_i)}, C_{i,(1-y_i)}$ per $\pi_{\mathcal{F}}$ in $\mathcal{G}$; and (4) the security Yao's construction.

3.3 Two Rounds are Necessary

It is almost immediate that two rounds are necessary for two-party computation under any reasonable definition of security. Loosely speaking, consider a candidate single-round protocol for a functionality that provides output to one of the parties, say P_2. Since (an honest) P_1 sends its message independently of P_2's input, P_2 can (honestly) run its output-computation side of the protocol on the incoming message multiple times using inputs of its choice, and learn the output of the functionality on each. This clearly violates security except for functions that do not depend on P_2's input.

More formally and in the context of UC security, consider the functionality $\mathcal{F}_{=}$, which on input a pair of two-bit strings $x, y \in \{0,1\}^2$, provides P_2 with output 1 if $x = y$, 0 otherwise. Assume π UC realizes $\mathcal{F}_{=}$ in a single round. Let π^{P_1} be the procedure in π that takes P_1's input x and a security parameter k and outputs P_1's outgoing message m; let π^{P_2} be the procedure in π that takes P_2's input y, an incoming message m and a security parameter k, and computes P_2's output value v. As π UC realizes $\mathcal{F}_{=}$, it must be the case that for any x, y and with all but negligible probability in k, if $m \overset{R}{\leftarrow} \pi^{P_1}(x, k)$ and $v \overset{R}{\leftarrow} \pi^{P_2}(y, m, k)$, then $v = \mathcal{F}_{=}(x, y)$ (by considering a benign adversary that does not corrupt any party and delivers all messages as prescribed by π).

Consider an environment $\mathcal{Z}$ which picks x uniformly at random from $\{0,1\}^2$ and provides x as input to P_1. Consider an adversary $\mathcal{A}$, participating in a real-life execution of π, that acts as follows. $\mathcal{A}$ corrupts P_2 on the onset of the execution. On an incoming message m from P_1, $\mathcal{A}$ computes $\pi^{P_2}(y, m, k)$ on *all* four strings $y \in \{0,1\}^2$, and outputs (the lexicographically first) y on which the computation produces 1. Note that by the above, with all but negligible probability, $\mathcal{A}$ outputs x. We claim that for any ideal-process adversary $\mathcal{S}$, $\mathcal{Z}$ may distinguish a real-life execution of π in the presence of $\mathcal{A}$ from the ideal process involving $\mathcal{S}$ and $\mathcal{F}_{=}$. To see this, observe that $\mathcal{S}$'s probability of outputting x is at most 1/4, as its view in the ideal process is independent of x.

4 Two-Round Universally-Composable Blind Signatures

In this section, we briefly discuss how our work can be used to construct a round-optimal (i.e., two-round) UC-secure blind signature scheme in the CRS model. We begin with a quick recap of the definitions. Roughly speaking, a blind signature scheme should guarantee *unforgeability* and *blindness*. The first requires that if a malicious user interacts with the honest signer for a total of ℓ executions of the protocol (in an arbitrarily-interleaved fashion), then the user should be unable to output valid signatures on $\ell+1$ distinct messages. (A stronger requirement called *strong unforgeability* requires that the user cannot even output $\ell+1$ distinct signatures on $\ell+1$ possibly-repeating messages.) Blindness requires, very informally, that a malicious signer cannot "link" a particular execution of the protocol to a particular user *even after observing the signature obtained by the user.* This is formalized (see, e.g., [Fis06]) by a game in which the signer interacts with two users in an order determined by a randomly-chosen selector bit b, and should be unable to guess the value of b (with probability significantly better than 1/2) even after being given the signatures computed by these two users. This definition also allows the malicious signer to generate its public key in any manner (and not necessarily following the legitimate key-generation algorithm).

The above represent the "classical" definitions of security for blind signatures. Fischlin [Fis06] formally defines a *blind signature functionality* in the UC framework. He also gives a two-round protocol realizing this functionality. Interestingly, one of the motivations cited in [Fis06] for *not* relying on the generic results of [CLOS02] is the desire to obtain a round-optimal protocol.

Assume we have a (standard) signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$, and consider the (randomized) functionality $f_{\mathsf{sign}}(SK, m) = \mathsf{Sign}_{SK}(m)$. Contrary to what might be a naive first impression, secure computation of this functionality does *not* (in general) yield a secure blind signature scheme! (See also [JL97].) Specifically, the problem is that the signer may use *different* secret keys SK, SK' in different executions of the protocol. Furthermore, the public key may be set up in such a way that each secret key yields a valid signature. Then, upon observing the signatures computed by the users, the signer may be able to tell which key was used to generate each signature, thereby violating the users' anonymity.

Juels, Luby, and Ostrovsky [JL97] suggest a relatively complex method for handling this issue. We observe that a much simpler solution is possible by simply forcing the signer to use *a fixed signing key in every execution of the protocol.* This is done in the following way: To generate a public/secret key, the signer first computes $(PK, SK) \leftarrow \mathsf{Gen}(1^k)$. It then computes a (perfectly-binding) commitment $\mathsf{com} = \mathsf{Com}(SK; \omega)$ to SK using randomness ω. The public key is PK, com and the secret key contains SK and ω.

Define functionality $f^*_{\mathsf{sign}}((SK, \omega), (\mathsf{com}, m))$ as follows: if $\mathsf{Com}(SK; \omega) = \mathsf{com}$, then the second party receives output $\mathsf{Sign}_{SK}(m)$ (when Sign is randomized, the functionality chooses a uniform random tape for computing this signature). Otherwise, the second party receives output $\perp$. The first party receives no output in either case.

It is not hard to see that a protocol for secure computation of f^*_{sign} yields a secure blind signature scheme (a proof is omitted); using a UC two-party computation protocol for f^*_{sign} gives a UC blind signature scheme. Using the simple two-round protocol constructed in Sect. 3.1, and noticing that only one party receives output here, we thus obtain a two-round UC blind signature scheme.

Acknowledgments

We are extremely grateful to the anonymous reviewers from Eurocrypt 2007 who suggested a way to recast our result in the setting where both parties communicate in a given round.

References

[AIR01] W. Aiello, Y. Ishai and O. Reingold. Priced Oblivious Transfer: How to Sell Digital Goods. In *Advances in Cryptology — EUROCRYPT 2001*, LNCS 2045, pp. 119–135, 2001.

[BCNP04] B. Barak, R. Canetti, J.B. Nielsen, and R. Pass. Universally Composable Protocols with Relaxed Set-Up Assumptions. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 186–195, 2004.

[BHS04] R. Bradshaw, J. Holt, and K. Seamons. Concealing complex policies with hidden credentials. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS)*, pp. 146–157, 2004.

[BMC06] W. Bagga, R. Molva, and S. Crosta. Policy-Based Encryption Schemes from Bilinear Pairings. In$\tilde{\mathrm{P}}$roceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS), p. 368, 2006.

[BMR90] D. Beaver, S. Micali, and P. Rogaway. The Round Complexity of Secure Protocols. In *Proceedings of the 22nd ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 503–513, 1990.

[Can01] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 136–145, 2001. Full version available at http://eprint.iacr.org/2000/067.

[Cha82] D. Chaum. Blind Signatures for Untraceable Payments. In *Advances in Cryptology — CRYPTO 1982*, LNCS, pp. 199–203, 1982.

[CCKM00] C. Cachin, J. Camenisch, J. Kilian, and J. Müller. One-Round Secure Computation and Secure Autonomous Mobile Agents. In *27th International Colloquium on Automata, Languages and Programming (ICALP)*, LNCS 1853, pp. 512–523, 2000.

[CDPW07] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally Composable Security with Global Setup. In *Theory of Cryptography — TCC 2007*, LNCS 4392, pp. 61–85, 2007.

[CF01] R. Canetti and M. Fischlin. Universally Composable Commitments. In *Advances in Cryptology — CRYPTO 2001*, LNCS 2139, pp. 19–40, 2001.

[CKL06] R. Canetti, E. Kushilevitz, and Y. Lindell. On the Limitations of Universally Composable Two-Party Computation Without Set-Up Assumptions. In *Journal of Cryptology*, vol. 19(2), pp. 135–167, 2006.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. In *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 494–503, 2002. Full version available at http://eprint.iacr.org/2002/140.

[DDO$^+$01] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano and A. Sahai. Robust Non-Interactive Zero-Knowledge. In *Advances in Cryptology — CRYPTO 2001*, LNCS 2139, pp. 566–598, 2001.

[DH76] W. Diffie and M. Hellman. New Directions in Cryptography. In *IEEE Transactions on Information Theory*, vol. 22(6), pp. 644–654, 1976.

[Fis06] M. Fischlin. Round-Optimal Composable Blind Signatures in the Common Reference String Model. In *Advances in Cryptology — CRYPTO 2006*, LNCS 4117, pp. 60–77, 2006.

[FAL06] K. B. Frikken, M. J. Atallah, and J. Li. Attribute-Based Access Control with Hidden Policies and Hidden Credentials. In *IEEE Transactions on Computers*, vol. 55(10), pp. 1259–1270, 2006.

[FGG$^+$06] M. Fitzi, J. Garay, S. Gollakota, C.P. Rangan, and K. Srinathan. Round-Optimal and Efficient Verifiable Secret Sharing. In *Theory of Cryptography — TCC 2006*, LNCS 3876, pp. 329–342, 2006.

[FL82] M.J. Fischer and N.A. Lynch. A Lower Bound for the Time to Assure Interactive Consistency. In *Information Processing Letters*, vol. 14(4), pp. 183–186, 1982.

[FM97] P. Feldman and S. Micali. An Optimal Probabilistic Protocol for Synchronous Byzantine Agreement. In *SIAM Journal of Computing*, vol. 26(4), pp. 873–933, 1997.

[Gol01] O. Goldreich. Foundations of Cryptography: Volume 1 – Basic Tools. Cambridge University Press, 2001.

[Gol04] O. Goldreich. Foundations of Cryptography: Volume 2 – Basic Applications. Cambridge University Press, 2004.

[GIKR01] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The Round Complexity of Verifiable Secret Sharing and Secure Multicast. In *Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 580–589, 2001.

[GIKR02] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. On 2-Round Secure Multiparty Computation. In *Advances in Cryptology — CRYPTO 2002*, LNCS 2442, pp. 178–193, 2002.

[GK96a] O. Goldreich and A. Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. In *Journal of Cryptology*, vol. 9(3), pp. 167–190, 1996.

[GK96b] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. In *SIAM Journal of Computing*, vol. 25(1), pp. 169–192, 1996.

[GM98] J. Garay and Y. Moses. Fully Polynomial Byzantine Agreement for $n > 3t$ Processors in $t + 1$ Rounds. In *SIAM Journal of Computing*, vol. 27(1), pp. 247–290, 1998.

[GMR89] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *SIAM J. of Computing*, vol. 18(1), pp. 186–208, 1989.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game, or A Completeness Theorem for Protocols with Honest Majority. In *Proceedings of the 19th ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 218–229, 1987.

[Hor07] O. Horvitz. Expressiveness of Definitions and Efficiency of Constructions in Computational Cryptography. Ph.D. thesis, University of Maryland, 2007.

[HMU05] D. Hofheinz, J. Müller-Quade, and D. Unruh. Universally Composable Zero-Knowledge Arguments and Commitments from Signature Cards. In *Proceedings of the 5th Central European Conference on Cryptology — MoraviaCrypt*, 2005.

[IK00] Y. Ishai and E. Kushilevitz. Randomizing Polynomials: A New Representation with Applications to Round-Efficient Secure Computation. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 294–304, 2000.

[JL97] A. Juels, M. Luby, and R. Ostrovsky. Security of Blind Digital Signatures. In *Advances in Cryptology — CRYPTO 1997*, LNCS 1294, pp. 150–164, 1997.

[JS07] S. Jarecki and V. Shmatikov. Efficient Two-Party Secure Computation on Committed Inputs. In *Advances in Cryptology — EUROCRYPT 2007*, LNCS 4515, 2007.

[Katz07] J. Katz. Universally Composable Multi-Party Computation using Tamper-Proof Hardware. In *Advances in Cryptology — EUROCRYPT 2007*, LNCS 4515, 2007.

[KO04] J. Katz and R. Ostrovsky. Round-Optimal Secure Two-Party Computation. In *Advances in Cryptology — CRYPTO 2004*, LNCS 3152, pp. 335–354, 2004.

[KOS03] J. Katz, R. Ostrovsky, and A. Smith. Round Efficiency of Multi-Party Computation with a Dishonest Majority. In *Advances in Cryptology — EUROCRYPT 2003*, LNCS 2656, pp. 578–595, 2003.

[Lin01] Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. In *Journal of Crypto*, vol. 16(3), pp. 143–184, 2003.

[LDB03] N. Li, W. Du, and D. Boneh. Oblivious Signature-Based Envelope. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 182–189, 2003.

[LL06] J. Li and N. Li. A Construction for General and Efficient Oblivious Commitment Based Envelope Protocols. In *Proceedings of 8th International Conference on Information and Communications Security (ICICS)*, pp. 122–138, 2006.

[LP04] Y. Lindell and B. Pinkas. A Proof of Yao's Protocol for Secure Two-Party Computation. In *Journal of Cryptology*, to appear. Full version available at http://eprint.iacr.org/2004/175.

[NP01] M. Naor and B. Pinkas. Efficient Oblivious Transfer Protocols. In *Proceedings of the 12th Symposium on Discrete Algorithms (SODA)*, pp. 448–457, 2001.

[NT05] S. Nasserian and G. Tsudik. Revisiting Oblivious Signature-Based Envelopes. Available at http://eprint.iacr.org/2005/283.

[PSL80] M. Pease, R. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. In *Journal of the ACM*, vol. 27(2), pp. 228–234, 1980.

[PW00] B. Pfitzmann and M. Waidner. Composition and Integrity Preservation of Secure Reactive Systems. In *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS)*, pp. 245–254, 2000.

[SYY99] T. Sander, A. Young, and M. Yung. Non-Interactive CryptoComputing For NC^1. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 554–567, 1999.

[Tau05] Y. Tauman Kalai. Smooth Projective Hashing and Two-Message Oblivious Transfer. In *Advances in Cryptology — EUROCRYPT 2005*, LNCS 3494, pp. 78–95, 2005.

[Yao86] A.C.-C. Yao. How to Generate and Exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 162–167, 1986.