Improved OR Composition of Sigma-Protocols

Michele Ciampi
DIEM
Università di Salerno
ITALY
mciampi@unisa.it

Giuseppe Persiano
DISA-MIS
Università di Salerno
ITALY
giuper@gmail.com

Alessandra Scafuro
Boston University and
Northeastern University
USA
scafuro@bu.edu

Luisa Siniscalchi
DIEM
Università di Salerno
ITALY
lsiniscalchi@unisa.it

Ivan Visconti
DIEM
Università di Salerno
ITALY
visconti@unisa.it

Abstract

In [CDS94] Cramer, Damgård and Schoenmakers (CDS) devise an OR-composition technique for Σ-protocols that allows to construct highly-efficient proofs for compound statements. Since then, such technique has found countless applications as building block for designing efficient protocols.

Unfortunately, the CDS OR-composition technique works only if *both* statements are fixed before the proof starts. This limitation restricts its usability in those protocols where the theorems to be proved are defined at different stages of the protocol, but, in order to save rounds of communication, the proof must start even if not all theorems are available. Many round-optimal protocols ([KO04, DPV04, YZ07, SV12]) crucially need such property to achieve round-optimality, and, due to the inapplicability of CDS's technique, are currently implemented using proof systems that requires expensive NP reductions, but that allow the proof to start even if no statement is defined (a.k.a., LS proofs from Lapidot-Shamir [LS90]).

In this paper we show an improved OR-composition technique for Σ-protocols, that requires only one statement to be fixed when the proof starts, while the other statement can be defined in the last round. This seemingly weaker property is sufficient for the applications, where typically one of the theorems is fixed before the proof starts. Concretely, we show how our new OR-composition technique can directly improve the round complexity of the efficient perfect quasi-polynomial time simulatable argument system of Pass [Pas03] (from four to three rounds) and of efficient resettable WI arguments (from five to four rounds).

Contents

1 Introduction

Witness-indistinguishable (WI) proofs. WI[1] proofs are fundamental for the design of cryptographic protocols, particularly when they are also proofs of knowledge (PoK). In a WIPoK the prover $\mathcal{P}$ proves knowledge of a witness certifying the veracity of a statement $x \in L$ to a verifier $\mathcal{V}$. WIPoKs can be used directly in some applications (e.g., in identification schemes) or can be a building block for stronger security notions (e.g., for zero-knowledge proofs using the FLS [FLS90] paradigm or for round-optimal secure computation [KO04]).

Round complexity of cryptographic protocols has been extensively studied both for its practical relevance and for its natural and conceptual interest. Regarding WIPoKs, we know from Blum's protocol [Blu86] that 3-round WIPoKs exist for all NP languages under the sole assumptions that one-way permutations exist. This result is obtained by designing a WIPoK for the language of Hamiltonian graphs and then by leveraging on the NP-completeness of the language of Hamiltonian graphs. Under stronger cryptographic assumptions, 2-round WI proofs, called ZAPs, and non-interactive WI (NIWI) proofs have been shown in [DN00, GOS06, BP15]. Neither ZAPs nor NIWI proofs are PoKs.

Since NP reductions are extremely expensive, several practical interactive PoKs have been designed for languages that are used in real-world cryptographic protocols (e.g., for proving knowledge of a discrete logarithm (DLog)). The study of such ad-hoc protocols mainly concentrates on a standardized form of a 3-round PoK referred to as Σ-protocol [Dam10, Sch89].

Σ-protocols. A Σ-protocol for an NP language L with witness relation R_L is a 3-round proof system jointly run by a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ in which $\mathcal{P}$ proves knowledge of a witness w for $x \in L$. In a Σ-protocol the only message sent by $\mathcal{V}$ is a random string. Such proof systems have two very useful properties: special soundness, which is a strong form of proof of knowledge, and special honest-verifier zero knowledge (SHVZK). The latter property basically says the following: if the challenge is known in advance, then by just knowing also the theorem, it is possible to generate an accepting transcript without using the witness. This is formalized through the existence of a special simulator, called the SHVZK simulator that, on input *a theorem* x and a challenge c, will output (a, z) such that (a, c, z) is an accepting 3-message transcript for x and is indistinguishable from the transcript produced by the honest prover when the challenge is c. Blum's protocol for Graph Hamiltonicity is an example of a Σ-protocol. Another popular example of Σ-protocols is Schnorr's protocol [Sch89] for proving knowledge of a discrete logarithm.

The security provided by the SHVZK property is clearly insufficient as it gives no immediate guarantees against verifiers who deviates from the protocol. Despite of this, the success of Σ-protocols and their impact in various constructions [Lin15, CPSV15, CPSV16, LP15, CG15, GK15, ORV14, AOS13, SV12, OPV10, BPSV08, CV07, CDV06, Vis06, GMY06, CV05a, CV05b, DG03, BFGM01, PS96] is a fact. This is due to a breakthrough of Cramer et al. [CDS94] that adds WI to the security of Σ-protocol.

OR composition of Σ-protocols. Let L be a language that admits a Σ-protocol Π_L. In [CDS94] it is shown how to use Π_L and its properties to construct a new Σ-protocol, Π_L^{OR}, for proving the OR composition of theorems in L *avoiding* the NP reduction by crucially exploiting the honest-verifier zero-knowledge (HVZK[2]) property of Π_L. The rationale behind the transformation can

[1] We will use WI to mean both "witness indistinguishability" and "witness indistinguishable".

[2] HVZK requires the existence of a simulator that by receiving in input the theorem gives in output an accepting triple (a, c, z). Clearly HVZK is implied by SHVZK.

be informally explained as follows. The prover wishes to prove a statement of the form $((x_0 \in L) \vee (x_1 \in L))$. The naïve idea of simply running Π_L twice in parallel would not work because the prover knows only one of the witnesses, say w_b, and cannot compute two accepting transcripts without knowing w_{1-b}. However, due to the HVZK property, the prover can generate an accepting transcript for $x_{1-b} \in L$ even without knowing w_{1-b}, by running the HVZK simulator Sim associated with Π_L. Indeed, Sim "only" needs in input the theorem x_{1-b} and will output the entire transcript, challenge included. The trick is then to generate the challenges for the two executions of Π_L, in such a way that the prover can control the challenge of exactly one of them (but not both), and set it to the value generated by Sim. Note that, if running the algorithm of Sim is as efficient as running the algorithm of $\mathcal{P}$, then the composed protocol is efficient. We stress that this OR-composition technique preserves SHVZK and will refer to it as the CDS-OR technique.

A very interesting property of this transformation, besides the fact that it does not need NP reduction, is that if Sim is a simulator for perfect HVZK then Π_L^{OR} is WI (this was shown in [CDS94]). This result was further extended by Garay et al. [GMY06] that noted that the CDS-OR technique can be used also for Σ-protocols that are computational HVZK. In this case the relation proved is slightly different, namely, starting with a relation $\mathcal{R}_L$ and instances x_0 and x_1, the resulting Π_L^{OR} protocol is computational WI for the relation

$$\mathcal{R}_L^{\mathsf{OR}} = \{((x_0, x_1), w) : ((x_0, w) \in \mathcal{R}_L \wedge (x_1 \in L)) \vee ((x_1, w) \in \mathcal{R}_L \wedge (x_0 \in L))\}.$$

Input-delayed proofs. Often in cryptographic protocols there is a preamble phase that has the purpose of establishing, at least in part, a statement to be proven with a WI proof. In such cases, since one of the statements is fully specified only when the preamble is completed, the WI proof can start only after the preamble ends. Hence, the overall round complexity of protocols that follow this paradigm amounts to the sum of the round complexity of the preamble and of the WI proof.

In [LS90], Lapidot and Shamir (and later on Feige et al. in [FLS90]) show a 3-round proof of knowledge for Hamiltonian Graphs which has the special property that a prover can compute the first round of the proof, *without* knowing the theorem to be proved (that is, the graph) but only needs to know its size (that is, the number of vertices). Such a 3-round protocol is a Σ-protocol (and thus satisfies the SHVZK property) and is a WI proof. We will refer to this protocol as LS. Also, we will call *input delayed* a Σ-protocol where the prover computes the first message without knowledge of the statement to be proved.

The input-delayed property directly improves the round complexity of all the cryptographic protocols that follow the paradigm described above. The reason is that now the WI proof can start even if the preamble that generates the statement is not completed yet. It is worthy to note that in many applications the preamble serves as a mean to generate some trapdoor theorem, that is used only in the security proof. The "honest" theorem instead is typically known already at the beginning of the protocol. This technique has been used extensively and, most notably, it led to the celebrated FLS paradigm that upgrades any WI proof system into a zero-knowledge (ZK) proof system.

The input-delayed property of LS has been instrumental to provide round-efficient constructions from general assumptions, such as: 4-round (optimal) secure 2PC where only one player gets the output (5 rounds when both players get the output) [KO04], 4-round resettable WI arguments [YZ07, SV12], 4-round (optimal) resettable ZK for NP in the BPK model [YZ07, SV12].

Despite being so influential to achieve round efficiency for cryptographic protocols, the power of LS unfortunately vanishes as soon as practical constructions are desired. Indeed, similarly to

Blum's protocol, LS is crucially based on specific properties of Hamiltonian graphs. Thus, when used to prove more natural languages, which is the case of most of the applications using WI proofs, it requires to perform rather inefficient NP reductions.

Efficient protocols and limits of the CDS-OR technique. A natural question is what happens if we want to avoid the NP reduction and we try to use the CDS-OR technique to construct input-delayed adaptive WI proofs. A bit more specifically, we know that there exist Σ-protocols that are input delayed. Schnorr's protocol [Sch89] for DLog is such an example since the first message can be computed without knowing the instance, but only a group generator. Thus the question is what happens if we apply the CDS-OR technique to an input-delayed Σ-protocol. Do we obtain a WI Σ-protocol that is input delayed as well?

Unfortunately, the answer is negative. The CDS-OR technique does *not* preserve the input-delayed property, not even when used to compose two Σ-protocols that are both input delayed. To see why, recall that the CDS-OR composition technique when applied to Σ-protocol Π_L for language L requires the prover to compute two accepting transcripts, one of which is computed by running the HVZK simulator Sim. Recall that Sim needs in input the theorem to be proved. Hence, to prove knowledge of a witness for the compound theorem $(x_0 \in L \vee x_1 \in L)$, the prover, who knows one witness, say w_b, needs to know also x_{1-b} already at the first round to be able to run the simulator. Thus, in the CDS-OR technique the prover can successfully complete the protocol if and only if *both*[3] instances are specified already at the first round.

Because of this missing feature, the CDS-OR technique has limited power in allowing one to obtain round-efficient/optimal cryptographic protocols, compared to the number of rounds obtained by using LS. As such, in some cases when focusing on efficient constructions, the *best* round-complexity that we can achieve using efficient Σ-protocols and avoiding NP reductions needs at least one additional round, therefore requiring at least 5-round if one wants to match the previously mentioned applications (e.g., 5-round resettable ZK for NP in the $\mathtt{BPK}$ model [YZ07, SV12] and 5-round resettable WI [YZ07, SV12]) argument systems.

Additionally, we note that the CDS-OR technique is the bottleneck in the round-complexity of the 4-round straight-line perfect simulatable in quasi-polynomial time argument shown by Pass in [Pas03]. This argument uses quasi-polynomial time simulation and, potentially, it would only need three rounds as any Σ-protocol. The additional first round is required precisely to define the trapdoor theorem. Hence, the following natural question arises:

> *Given a language L with an input-delayed Σ-protocol Π_L, is it possible to design an efficient Witness Indistinguishable Σ-protocol Π^L_{OR} for proving knowledge of a witness certifying that $(x_0 \in L \vee x_1 \in L)$ that does* not *require knowledge of both x_0 and x_1 to play the first round?*

1.1 Our Contribution

In this paper we answer the above question positively for a large class of Σ-protocols that includes *all* Σ-protocols used in efficient constructions. Specifically, we propose a new OR-composition technique for Σ-protocols that relaxes the need of having both instances fixed before the Σ-protocol starts. Our technique allows the composition of Σ-protocols for different languages and leads to improved round complexity in previous efficient constructions based on CDS-OR technique.

[3]To see why, note that the WI property requires that the prover would be able to prove any of the two theorems, and thus potentially use the simulator on either x_0 or x_1.

Namely, we describe the following two results that we obtain by making use of our new OR-composition technique:

- Efficient 3-round straight-line perfect quasi-polynomial time simulatable argument system for a large class of useful languages. The previous construction required four rounds [Pas03].
- Efficient 4-round rWI argument system. Previous constructions required five rounds [YZ07, SV12].

Our new technique can also be used to replace LS towards obtaining efficient round-optimal resettable zero-knowledge arguments in the BPK model (using the constructions of [YZ07, SV12]), round-optimal secure two-party computation (using the construction of [KO04]) and 4-round non-malleable commitments (using the construction of [GRRV14]).

Finally, we provide a precise classification of the Σ-protocols that can be used in our new OR-composition technique. In the following paragraphs we first provide a high-level description our OR-composition technique, then we discuss the applications in more details.

1.2 Our Techniques

Overview. We start by defining the setting we are considering. Let L_0 and L_1 be any pair of languages admitting Σ-protocols Π_0 and Π_1. We want to construct a Σ-protocol Π_L^{OR} for the language $L = L_0 \vee L_1$. An instance of L is a pair (x_0, x_1) and we want only x_0 to be specified before Π_L^{OR} starts while x_1 is specified only upon the last round of the protocol[4]. We assume that Π_1 is an *input-delayed* Σ-protocol and thus the first prover message of Π_1 can be computed without knowing x_1. As mentioned earlier this property is satisfied by popular Σ-protocols such as the ones for Discrete Log, Diffie-Hellman triples, and of course, LS itself.

Now, recall that the problem with the CDS-OR technique was that a prover needs to run Sim to compute the first round of the protocol, and this necessarily requires knowledge of *both* theorems before the protocol starts. We want instead that the prover uses only knowledge of x_0.

We solve this problem by introducing a new OR-composition technique that does not require the prover to run Sim on x_1 already in the first round. Instead, our technique allows the prover to wait and take action only in the third round when x_1 is finally defined.

Our starting point is the well known fact that given any Σ-protocol there exists an instance-dependent trapdoor commitment (IDTC) scheme where the witness for the membership of the instance in the language can be used as a trapdoor to open a committed message as any desired message, as in [DG03]. Our next observation is that, instead of having the prover send the first round for protocol Π_1 in the clear, we can have him send a commitment to it, and such commitment can be computed using an instance-dependent trapdoor commitment based on Π_0 with respect to instance x_0. Recall that this is possible, as in our setting we assume that Π_1 is an input-delayed Σ-protocol, so the prover can honestly compute the first message of Π_1 without knowing x_1. Therefore, the first round of our Π_L^{OR} protocol, is simply an IDTC of a honest Π_1's first round.

Later on, upon receiving the challenge c from the verifier, and after the theorem x_1 is defined, the prover computes the third round as follows. If she has received a witness for x_0, then she will run Sim on input (x_1, c) to compute an accepting transcript of Π_1 for x_1. Then, using the witness w_0 she will equivocate the commitment sent in the first round, according to the message output

[4]Like LS, we will just need the size of x_1 to be known when Π_L^{OR} starts.

by Sim. Otherwise, if she has received a witness for x_1 then she does not need to equivocate: she will honestly open the commitment, and honestly compute the third message of Π_1. Therefore, the third round of Π_L^{OR}, simply consists of an opening of the IDTC together with the third message of Π_1.

Now note that this idea works only if we have a special IDTC scheme that has the following strong trapdoor property: a sender can equivocate even a commitment that has been computed honestly. Unfortunately, this property is not satisfied in general by any trapdoor commitment based on Σ-protocols, but only for some. This would restrict the class of Σ-protocols that we can use as L_0 in our technique. For example, this class would not contain Blum's protocol.

Our next contribution is the construction of IDTC schemes that satisfy this strong trapdoor property, for a large class of Σ-protocols. Towards this goal, we define the notion of a t-IDTC scheme which are IDTCs for which the ability to open a commitment in t ways implies knowledge of a witness for the instance associated with the commitment. Next, we construct 2-IDTC and 3-IDTC schemes based on two different classes of Σ-protocols, the union of which includes all the Σ-protocols that are commonly used in cryptographic protocols. Finally, we provide a general OR-composition technique for any pair of languages L_0 and L_1 such that L_0 has a t-IDTC scheme and L_1 has an input-delayed Σ-protocol.

t-instance-dependent trapdoor commitment scheme. For integer $t \geq 2$, a t-IDTC scheme for a polynomial-time relation $\mathcal{R}$ admitting Σ-protocol $\Pi_{\mathcal{R}}$ is a triple $(\mathsf{TCom}, \mathsf{TDec}, \mathsf{TFake})$ where TCom, TDec are the honest commitment/decommitment procedures and TFake is the equivocation procedure that, given a witness for an instance x, equivocates any commitment with respect to x computed by TCom. The crucial differences between a t-IDTC scheme and a regular trapdoor commitment scheme are: (a) the trapdoor property is strong in the sense that knowledge of the trapdoor (that is, the witness of the instance x) allows to equivocate even commitments that have been honestly computed; (b) the binding property is relaxed: in a t-IDTC scheme, the sender can open the same commitment in $t-1$ different ways, even without the trapdoor. This relaxation allows us to build an IDTC scheme from a wider class of Σ-protocols, which will cover all the Σ-protocols that have been used in literature.

Constructing a 2-IDTC scheme. A 2-IDTC scheme can be straight-forwardly constructed from any Σ-protocol Π_0 that has the following property: even if the first message a_0 was computed by the SHVZK simulator Sim, an accepting z_0 can be efficiently computed, for every challenge c_0, by using knowledge of the witness and of the randomness used by Sim to produce a_0. We call the Σ-protocols that satisfy this property, *chameleon* Σ-protocols, and we denote by $\mathsf{P}_{\mathsf{sim}}$ the special prover strategy that can answer any challenge even starting from a simulated a_0.

More precisely, given a chameleon Σ-protocol Π_0 for a language L_0, one can construct a 2-IDTC scheme as follows. Let $x_0 \in L_0$. To commit to a message m, the sender runs $\mathsf{Sim}(x_0, m; r_0)$ and obtains a_0, z_0. The commitment is the value a_0. The opening is the pair m, z_0. The commitment is accepted iff (x_0, a_0, m, z_0) is accepting. To equivocate a_0, as a message m', run the special prover algorithm $\mathsf{P}_{\mathsf{sim}}((x_0, m, r_0), w_0, m')$ and obtain an accepting z_0.

Constructing a 3-IDTC scheme. We now discuss a different committing strategy that works for Σ-protocols in which the simulated first message a_0 can only be continued for the challenge specified by Sim, even if a witness is made available. Blum's protocol for Hamiltonicity is an example of a Σ-protocol with this property.

To commit to m, the sender sends a pair (a_0, a_0') where, with probability $1/2$, a_0 is obtained by running $\mathsf{Sim}(x_0, m)$ while a'_0 is computed by running the prover of Π_0, and with probability $1/2$ the above order is inverted. One can think of a commitment as composed of two threads: a *simulated* thread and a *honest* thread. To open the commitment, the prover sends m and z^*, and the verifier accepts the decommitment if m, z^* are accepting for one of the threads; namely, the verifier checks that either (a_0, m, z^*) or (a'_0, m, z^*) is accepting for $x_0 \in L_0$. To equivocate (a_0, a'_0) to a message m', the sender simply continues the thread of the honest prover, using m' as challenge and computes z^* using the witness. Clearly, a malicious sender can open in two different ways even when $x_0 \notin L$. Nevertheless, three openings allow the extraction of the witness for x_0.

When our OR-composition technique is instantiated with a 3-IDTC scheme we have that the resulting protocol is still WI since no power is added to the verifier. However the protocol is *not* a Σ-protocol since the special-soundness property is not guaranteed. The reason is that, in a 3-IDTC scheme the sender can open the commitment in two different ways even without having the trapdoor (i.e., the witness for $x_0 \in L_0$). Therefore, for any challenge c sent by $\mathcal{V}$, the fact that the commitment of a_1 can be opened in two ways gives a malicious prover $\mathcal{P}^*$ two chances (a_1, c, z_1) and (a'_1, c, z'_1) to successfully complete the protocol for a false statement x_1. Nevertheless, this extra freedom does not hurt soundness as both openings (i.e., a_1 and a'_1) are fixed in advance, and thus when x_1 is not an instance of the language there exist only two challenges c' and c'' that would allow $\mathcal{P}^*$ to succeed. When the challenge is long enough the success probability of $\mathcal{P}^*$ is therefore negligible.

Our construction when starting from a 3-IDTC scheme is 3-special sound (i.e., answering to 3 challenges allows one to compute a witness efficiently), and therefore it is a proof of knowledge when the challenge is long enough.

1.3 Discussion

What really matters. Our new OR-composition technique works only when the theorem that has not been defined yet (i.e., x_1), admits an input-delayed Σ-protocol). We stress that this is not a limitation for the applications that we have in mind. In fact, in all *efficient* protocols that make use of input-delayed proofs that we are aware of, the preamble has always the purpose of generating the trapdoor theorem. In practical scenarios[5] L_1 usually corresponds to DLog or DDH. The fact that we can not have Blum's Σ-protocol for L_1 when L_1 is the language of Hamiltonian graphs, is therefore not relevant as the actual language of interest is L_0.

Comparison with the CDS-OR technique. We remark that even in the extremely simplified case where:

1. the two instances x_0 and x_1 are for the same language L,
2. L admits an *input-delayed* Σ-protocol Π_L which is also special HVZK,
3. Π_L is chameleon and thus one can compute the first message using Sim and then continue with the prover to answer to arbitrary challenges,
4. the prover knows in advance the witness w and instance x_b for which she will be able to honestly complete the protocol,

[5]These are the only scenarios of interest for our work since if practicality is not desired than one can just rely on the LS Σ-protocol and use NP reductions.

the CDS-OR technique fails in obtaining a Σ-protocol (or a WIPoK) for the OR composition of instances of L if any one of the instances is not known when the protocol starts.

Beyond Schnorr's protocol. The works of Cramer [Cra96], Cramer and Damgård [CD98], and Maurer [Mau09, Mau15] showed that a protocol (referred to as the *Pre-Image Protocol*) for proving knowledge of a pre-image of a group homomorphism unifies and generalizes a large number of protocols in the literature. Classic Σ-protocols, such as Schnorr's protocol [Sch89] and the Guillou-Quisquater protocol [GQ88], are particular cases of this abstraction. We show that the *Pre-Image Protocol* is a chameleon Σ-protocol and can thus be used in our construction.

What is in and what is out. As mentioned previously, the Σ-protocol for L_1 can be any *input-delayed* Σ-protocol. We now discuss which Σ-protocols can be used to instantiate L_0 in our OR transform. For this purpose, we identify four classes of Σ-protocols and we prove that any Σ-protocol that falls in any of the first three classes can be used in our OR transform (by instantiating either a 2-IDTC or a 3-IDTC scheme).

We also identify a class of Σ-protocols that is not suitable for any of our techniques. Luckily, we have no example of natural Σ-protocols that fall in this class, and in order to prove the separation we had to construct a very contrived scheme. The four classes are listed below.

- *(Class 1)* Σ-protocols that *are* Chameleon and *do not* require the witness to compute the first round. This class of Σ-protocols can be used to construct both 2-IDTC and 3-IDTC schemes.
- *(Class 2)* Σ-protocols that *are* Chameleon and *require* the prover to use the witness already to compute the first round. This class of Σ-protocols can be used to construct a 2-IDTC scheme.
- *(Class 3)* Σ-protocols that *are not* Chameleon but *do not require* the prover to use the witness in the first round. This class of Σ-protocols can be used to construct a 3-IDTC scheme.
- *(Class 4)* Σ-protocols that *are not* Chameleon and *require* the witness to be used already in the first round. This class of Σ-protocols can not be used in our techniques.

The input-delayed features. We stress here that our techniques allow to start and complete an efficient OR composition of two Σ-protocols (with the discussed restrictions) provided that one instance is known and another one will be known later. Having a witness for the first or the second instance always allows $\mathcal{P}$ to convince $\mathcal{V}$. This contrasts with the CDS-OR technique where knowing a witness for x_0 would block $\mathcal{P}$ immediately since $\mathcal{P}$ would need immediately x_1 to continue, but x_1 will not be available until the third round.

1.4 Applications

Our new OR-composition technique does not provide the full power of LS because it needs one theorem to be known before the protocol starts. However, as we show below, this seemingly weaker property suffices to improve the round-complexity of some of the previous constructions based on the CDS-OR technique. Such constructions aim to efficiently[6] transform a Σ-protocol for a relation $\mathcal{R}$ into a *round-efficient* argument with more appealing features.

[6] By *efficiently* we mean that no NP reduction is needed and only a constant number of modular exponentiations are added. We do not discuss the practicality of the resulting constructions.

Efficient 3-round straight-line perfect quasi-polynomial time simulatable argument system. We achieve this result directly, using the construction of Pass [Pas03] and replacing the CDS-OR technique with our technique. As a result the first round of the verifier of [Pas03] can be postponed, therefore reducing the round complexity from four to three rounds. Our construction works for all languages admitting a perfect chameleon Σ-protocol.

Efficient 4-round resettable WI arguments. It is well known [CGGM00] how to transform a Σ-protocol into a resettable WI protocol: the verifier commits to the challenge c using a perfectly hiding commitment scheme and sends it to the prover in the first round; the prover then computes its messages with randomness derived by applying a pseudo-random function (PRF) on the commitment received. Soundness follows directly from the soundness of the Σ-protocol due to the perfect hiding of the commitment. WI follows from the fact that the protocol is zero knowledge against a stand-alone verifier and thus concurrent WI. Then the use of the PRF and the fact that all messages of the verifier are committed in advance upgrades concurrent WI to resettable WI. This approach, however, generates a 5-round protocol.

Achieving the same result *efficiently*, namely, avoiding NP reductions, in only four rounds is non-trivial. The reason is that if we attempt to replace the 2-round perfectly hiding commitment with a non-interactive commitment, we lose the unconditional soundness property, and then it is not clear how to argue about computational soundness. More specifically, black-box extraction of the witness is not possible (black-box extraction and resettable WI can not coexist) and the adversarial prover could try to maul the commitment of the verifier and adaptively generate the first round of the Σ-protocol. In fact, even allowing complexity-leveraging arguments (and thus, straight-line extraction), constructing a 4-round WI argument system that avoids NP reductions and adds only a few modular exponentiations to the underlying Σ-protocol has remained so far an open problem.

We solve this problem by using our new OR-composition technique. We have the verifier commit to the challenge in the first round, but then later, instead of sending the decommitment, she will directly send the challenge and prove that either the challenge is the correct opening of the commitment or she solved some hard puzzle (in our construction, computing the Discrete Log of a random group element chosen by the prover). The puzzle is sent by the prover in the second round and it will be solved by the reduction in super-polynomial time in the proof of soundness.

This trick has been proposed in literature in various forms [Pas03, DPV04] and we are using the form used in [DPV04] where the puzzle is sent only in the second round. [DPV04] must use the LS transform and therefore needs NP-reduction. As explained earlier, going through LS *was* necessary as the CDS-OR transform can be applied only if both statements are fixed at the beginning.

Our new OR transform solves precisely this problem, and it allows the verifier to start the proof before the puzzle is defined, and this proof can be done efficiently without NP reductions.

Resettable WI follows from the CGGM transformation and the WI property of the proof generated by the prover. The groups used for the commitment of the challenge and for the puzzle sent by the prover, will be chosen appropriately so that the hardness of computing discrete logarithms are different and guarantee that our reductions work (i.e., we make use of complexity leveraging).

Further applications. Our new OR-composition technique can find various other applications. Indeed, wherever there is a round-efficient (but otherwise inefficient) construction based on the use of LS without a corresponding efficient construction with the *same* round complexity, then our technique constitutes a powerful tool towards achieving computationally efficient and round-efficient constructions. For instance, the 4-round (optimal) resettable ZK argument systems in the BPK model provided in [YZ07, SV12], consists (roughly) of the parallel execution of a (resettable) WI protocol from the prover to the verifier, where the prover proves that either $x \in L$ or he knows the secret key associated to the public identity of the verifier, and a 3-round (resettably-sound) WI protocol from the verifier to the prover, where $\mathcal{V}$ proves knowledge of the secret key associate to its public key, or knowledge of the solution of a puzzle computed by the prover. When instantiated with efficient Σ-protocols, such construction requires 5-rounds, where the additional round, from the prover to the verifier, is used to send the puzzle necessary for the verifier to start a proof using the CDS-OR technique. We observe that this setting closely resembles the setting of the 4-round resettable WI (rWI) protocol that we provide in this paper. As such, one could directly instantiate the proof provided by the prover of the BPK model, with our 4-round rWI protocol, and have the verifier just prove knowledge of its secret keys, thus avoiding the need of the additional first round.

Other applications where our new OR-composition technique could be useful consist in replacing the use of LS in the 4-round non-malleable commitment scheme of [GRRV14], and in the round-optimal secure two-party computation protocol of [KO04].

1.5 Open Problems

Our OR-composition technique relaxes the requirement of CDS-OR of requiring knowledge of *all* instances already at the beginning of the protocol. However still our result does not match the power of LS where *no* theorem is required for the protocol to start. An immediate open question is whether one can improve our OR transform so that the first round can be run without the knowledge of any theorem.

Perhaps a first step in this direction would be to answer a related relaxed question, which is to design an OR transform for proving (still preserving WI) knowledge of 1 out of n theorems and that requires knowledge of (at least some) theorems only after the second round.

It would also be interesting to extend our technique in order to make it applicable to *all* Σ-protocols.

2 Definitions

In this section we set-up our notation and review some standard definitions and assumptions that will be used in the paper.

We denote the security parameter by λ.

If A is a probabilistic algorithm then $A(x)$ denotes the probability distribution of the output of A when it receives x as input. By $A(x; R)$ instead we denote the output of A on input x when coin tosses R are used as randomness.

A *polynomial-time relation* $\mathcal{R}$ (or, simpy, a *relation*) is a subset of $\{0,1\}^\star \times \{0,1\}^\star$ for which membership of (x, w) to $\mathcal{R}$ can be decided in time polynomial in $|x|$. We define the NP-language $L_\mathcal{R}$ as $L_\mathcal{R} = \{x | \exists w : (x, w) \in \mathcal{R}\}$. If $(x, w) \in \mathcal{R}$, we say that w is a *witness* for *instance* x. Following [GMY06], we define $\hat{L}_\mathcal{R}$ to be the input language that includes both $L_\mathcal{R}$ and all well formed instances that do not have a witness. More formally, $L_\mathcal{R} \subseteq \hat{L}_\mathcal{R}$ and membership in $\hat{L}_\mathcal{R}$ can be tested in polynomial time. We implicitly assume that the verifier of a protocol for relation $\mathcal{R}$ executes the protocol only if the common input x belongs to $\hat{L}_\mathcal{R}$ and rejects immediately common inputs not in $\hat{L}_\mathcal{R}$.

For two interactive machines A and B, we denote by $\langle A(\alpha), B(\beta)\rangle(\gamma)$ the output of B after running on private input β with A using private input α, both running on common input γ.

2.1 Number-Theoretic Assumptions

We define *group generator* algorithms to be probabilistic polynomial-time algorithms that take as input security parameter 1^λ and output $(\mathcal{G}, q, g)$, where $\mathcal{G}$ is (the description of) a cyclic group of order q and g is a generator of $\mathcal{G}$. We assume that membership in $\mathcal{G}$ and its group operations can be performed in time polynomial in the length of q and that there is an efficient procedure to randomly select elements from $\mathcal{G}$. Moreover, with a slight abuse of notation, we will use $\mathcal{G}$ to denote the group and its description.

We consider the sub-exponential versions of the DLog and of the DDH assumptions that posit the hardness of the computation of discrete logarithms and of breaking the Decisional Diffie-Hellman assumption with respect to the group generator algorithm IG that, on input λ, randomly selects a λ-bit prime q such that $p = 2q + 1$ is also prime and outputs the order q group $\mathcal{G}$ of the quadratic residues modulo p along with a random generator g of $\mathcal{G}$. The strong versions of the two assumptions posit the hardness of the same problems even if p (and q) and generator g are chosen adversarially. More precisely:

Assumption 1 (DLog Assumption)**.** *There exists a constant α such that for every probabilistic algorithm A running in time 2^{λ^α} the following probability is a negligible function of λ*

$$\text{Prob}\left[\ (\mathcal{G}, q, g) \leftarrow \mathsf{IG}(1^\lambda); y \leftarrow \mathbb{Z}_q : A(g^y) = y\ \right].$$

Assumption 2 (Strong DLog Assumption [CD08])**.** *Consider a pair of probabilistic algorithms (A_0, A_1) such that A_0, on input 1^λ, outputs $(\mathcal{G}, q, g)$, where $\mathcal{G}$ is the group of the quadratic residues modulo p, where p is prime, $p = 2q + 1$, q is a λ-bit prime and $g \in \mathcal{G}$, along with some auxiliary information* $\mathtt{aux}$*. There exists a constant α such that for any such pair (A_0, A_1) running in time 2^{λ^α} the following probability is a negligible function of λ:*

$$\text{Prob}\left[\ ((\mathcal{G}, q, g), \mathtt{aux}) \leftarrow A_0(1^\lambda); y \leftarrow \mathbb{Z}_q : A_1(g^y, \mathtt{aux}) = y\ \right].$$

We next introduce the DDH Assumption and the Strong DDH Assumption which imply the DLog Assumption and the Strong DLog Assumption, respectively.

Assumption 3 (DDH Assumption)**.** *There exists a constant α such that, for every probabilistic algorithm A running in time 2^{λ^α}, the following is a negligible function of λ*

$$\Big|\text{Prob}\left[\ (\mathcal{G}, q, g) \leftarrow \mathsf{IG}(1^\lambda); x, y, z \leftarrow \mathbb{Z}_q : A((\mathcal{G}, q, g), g^x, g^y, g^z) = 1\ \right] - \text{Prob}\left[\ (\mathcal{G}, q, g) \leftarrow \mathsf{IG}(1^\lambda); x, y, z \leftarrow \mathbb{Z}_q : A((\mathcal{G}, q, g), g^x, g^y, g^{xy}) = 1\ \right]\Big|.$$

Assumption 4 (Strong DDH Assumption)**.** *Consider a pair of probabilistic algorithms (A_0, A_1) such that A_0, on input 1^λ, outputs $(\mathcal{G}, q, g)$, where $\mathcal{G}$ is the group of the quadratic residues modulo p, where p is prime, $p = 2q + 1$, q is a λ-bit prime and $g \in \mathcal{G}$, along with some auxiliary information* $\mathtt{aux}$*. There exists a constant α such that, for any such pair (A_0, A_1) running in time 2^{λ^α}, the following is a negligible function of λ*

$$\Big|\text{Prob}\left[\ ((\mathcal{G}, q, g), \mathtt{aux}) \leftarrow A_0(1^\lambda); x, y, z \leftarrow \mathbb{Z}_q : A_1((\mathcal{G}, q, g), g^x, g^y, g^z, \mathtt{aux}) = 1\ \right] - \text{Prob}\left[\ ((\mathcal{G}, q, g), \mathtt{aux}) \leftarrow A_0(1^\lambda); x, y, z \leftarrow \mathbb{Z}_q : A_1((\mathcal{G}, q, g), g^x, g^y, g^{xy}, \mathtt{aux}) = 1\ \right]\Big|.$$

3 Σ-Protocols

We consider *3-move protocols* Π for a polynomial-time relation $\mathcal{R}$. Protocol Π is played by a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ that receive a common input x. $\mathcal{P}$ receives as an additional private input a witness w for x and the security parameter 1^λ in unary. The protocol Π has the following form:

1. $\mathcal{P}$ executes algorithm P_1 on common input x, private input w, security parameter 1^λ and randomness R obtaining $a = \mathsf{P}_1(x, w, 1^\lambda; R)$ and sends a to $\mathcal{V}$.
2. $\mathcal{V}$, after receiving a from $\mathcal{P}$, chooses a random *challenge* $c \leftarrow \{0,1\}^l$ and sends c to $\mathcal{P}$.
3. $\mathcal{P}$ executes algorithm P_2 on input x, w, R, c and sends $z \leftarrow \mathsf{P}_2(x, w, R, c)$ to $\mathcal{V}$.
4. $\mathcal{V}$ executes and outputs $\mathsf{V}(x, a, c, z)$ (i.e., $\mathcal{V}$'s decision to accept $(b = 1)$ or reject $(b = 0)$).

We call $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$ the algorithms *associated* with Π and l the challenge length such that, wlog, the challenge space $\{0,1\}^l$ is composed of 2^l different challenges.

The triple (a, c, z) of messages exchanged is called a *3-move transcript.* A 3-move transcript is *honest* if a, z correspond to the messages computed running the honest algorithms, respectively, of P_1 and P_2, and c is a random string, in $\{0,1\}^l$. A 3-move transcript (a, c, z) is *accepting* for x if and only if $\mathsf{V}(x, a, c, z) = 1$. Two accepting 3-move transcripts (a, c, z) and (a', c', z') for an instance x constitute a *collision* if $a = a'$ and $c \neq c'$.

Definition 1 (Σ-protocol [CDS94])**.** *A 3-move protocol* Π *with challenge length* l *is a* Σ-protocol *for a relation* $\mathcal{R}$ *if it enjoys the following properties:*

1. ***Completeness****. If* $(x, w) \in \mathcal{R}$ *then all honest 3-move transcripts for* (x, w) *are accepting.*
2. ***Special Soundness****. There exists an efficient algorithm* $\mathsf{Extract}$ *that, on input* x *and a collision for* x*, outputs a witness* w *such that* $(x, w) \in \mathcal{R}$*.*
3. ***Special Honest-Verifier Zero Knowledge (SHVZK)****. There exists a PPT simulator algorithm* Sim *that takes as input* $x \in L_{\mathcal{R}}$*, security parameter* 1^λ *and* $c \in \{0,1\}^l$ *and outputs an accepting transcript for* x *where* c *is the challenge. Moreover, for all* l*-bit strings* c*, the distribution of the output of the simulator on input* (x, c) *is computationally indistinguishable from the distribution of the 3-move honest transcript obtained when* $\mathcal{V}$ *sends* c *as challenge and* $\mathcal{P}$ *runs on common input* x *and any private input* w *such that* $(x, w) \in \mathcal{R}$*.*

   We say that Π *is* Perfect *when the two distributions are identical.*

Not to overburden the descriptions of protocols and simulators, we will omit the specification of the security parameter when it is clear from the context.

In the rest of the paper, we will call a 3-move protocol that enjoys Completeness, Special Soundness and Honest-Verifier Zero Knowledge (HVZK[7]) a $\tilde{\Sigma}$*-protocol.* The next theorem shows that SHVZK can be added to a 3-move protocol with HVZK without any significant penalty in terms of efficiency.

[7]Recall that HVZK requires the existence of a simulator that generates a full transcript. This is a seemingly weaker requirement than SHVZK where the challenge is an input for the simulator.

Theorem 1 ([Dam10])**.** *Suppose relation $\mathcal{R}$ admits a 3-move protocol Π' that is HVZK (resp., perfect HVZK). Then $\mathcal{R}$ admits a 3-move protocol Π that is SHVZK (resp., perfect SHVZK) and has the same efficiency.*

Proof. Let l be the challenge length of Π', let $(\mathsf{P}'_1, \mathsf{P}'_2, \mathsf{V}')$ be the algorithms associated with Π' and let Sim' be the simulator for Π'. Consider the following algorithms.

1. P_1, on input $(x, w) \in \mathcal{R}$, security parameter 1^λ and randomness R_1, parses R_1 as (r_1, c'') where $|c''| = l$, computes $a' \leftarrow \mathsf{P}'_1(x, w, 1^\lambda; r_1)$, and outputs $a = (a', c'')$.
2. P_2, on input $(x, w) \in \mathcal{R}$, R_1 and randomness R_2 parses R_1 as (r_1, c''), c, sets $c' = c \oplus c''$, computes $z' \leftarrow \mathsf{P}'_2(x, w, r_1, c'; R_2)$, and sends it to $\mathcal{V}$.
3. V, on input x, $a = (a', c'')$, c and z', returns the output of $\mathsf{V}'(x, a', c \oplus c'', z')$ to decide whether to accept or not.

Consider the following PPT simulator Sim that, on input an instance x and a challenge c, runs Sim' on input x and obtains (a', c', z'). Then Sim sets $c'' = c \oplus c'$ and $a = (a', c'')$ and outputs (a, c, z'). It is easy to see that if Sim' is a HVZK (resp. perfect HVZK) simulator for Π' then Sim is a SHVZK (resp. perfect SHVZK) simulator for Π. $\square$

Definition 2 ([BG92, Dam10])**.** *Let $k : \{0,1\}^* \to [0,1]$ be a function. A protocol $(\mathcal{P}, \mathcal{V})$ is a* proof of knowledge *for the relation $\mathcal{R}$ with* knowledge error *k if the following properties are satisfied:*

- **Completeness** *If $\mathcal{P}$ and $\mathcal{V}$ follow the protocol on input x and private input w to $\mathcal{P}$ where $(x, w) \in \mathcal{R}$, then $\mathcal{V}$ always accepts.*
- **Knowledge soundness:** *There exists a constant $c > 0$ and a probabilistic oracle machine E, called the* extractor*, such that for every interactive prover $P^\star$ and every $x \in L_{\mathcal{R}}$, the machine E satisfies the following condition. Let $\epsilon(x)$ be the probability that $\mathcal{V}$ accepts on input x after interacting with $P^\star$. If $\epsilon(x) > k(x)$, then upon input x and oracle access to $P^\star$, the machine E outputs a string w such that $(x, w) \in \mathcal{R}$ within an expected number of steps bounded by*

$$\frac{|x|^c}{\epsilon(x) - k(x)}.$$

Theorem 2 ([Dam10])**.** *Let Π be a Σ-protocol for a relation $\mathcal{R}$ with challenge length l. Then Π is a proof of knowledge with knowledge error 2^{-l}.*

Definition 3 (Input-delayed Σ-protocol)**.** *A Σ-protocol $\Pi = (\mathcal{P}, \mathcal{V})$ with $\mathcal{P}$ running PPT algorithms $(\mathsf{P}_1, \mathsf{P}_2)$ is an input-delayed Σ-protocol if P_1 takes as input only the length of the common instance and P_2 takes as input the common instance x, the witness w, the randomness R_1 used by P_1 and the challenge c received from the verifier.*

Definition 4 (Witness-delayed Σ-protocol)**.** *A Σ-protocol $\Pi = (\mathcal{P}, \mathcal{V})$ for a relation $\mathcal{R}$ with associated algorithms $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$ is a witness-delayed Σ-protocol if P_1 takes as input only the common instance x.*

In a *Chameleon* Σ*-protocol*, the prover can compute the first message by using the simulator and thus knowing only the input but not the witness. Once the challenge has been received, the prover can compute the last message (thus completing the interaction) by using the witness w (which is thus used only to compute the last message) and the coin tosses used by the simulator to compute the first message.

Definition 5 (Chameleon Σ-protocol)**.** *A* Σ*-protocol* Π *for polynomial-time relation* $\mathcal{R}$ *is a* Chameleon Σ*-protocol if there exists an SHVZK simulator* Sim *and an algorithm* $\mathsf{P}_{\mathsf{sim}}$ *satisfying the following property:*

Delayed Indistinguishability*: for all pairs of challenges* c_0 *and* c_1 *and for all* $(x, w) \in \mathcal{R}$*, the following two distributions* $\{R \leftarrow \{0,1\}^{|x|^d}; (a, z_0) \leftarrow \mathsf{Sim}(x, c_0; R);\ z_1 \leftarrow \mathsf{P}_{\mathsf{sim}}((x, c_0, R), w, c_1) : (x, a, c_1, z_1)\}$ *and* $\{(a, z_1) \leftarrow \mathsf{Sim}(x, c_1) : (x, a, c_1, z_1)\}$ *are indistinguishable, where* Sim *is the Special HVZK simulator and* d *is such that* Sim*, on input an* λ*-bit instance, uses at most* λ^d *random coin tosses. If the two distributions above are identical then we say that delayed indistinguishability is perfect, and* Π *is a* Perfect *Chameleon* Σ*-protocol.*

We remark that a chameleon Σ-protocol Π has two modes of operations: the standard mode when $\mathcal{P}$ runs P_1 and P_2, and a *delayed* mode when $\mathcal{P}$ uses Sim and $\mathsf{P}_{\mathsf{sim}}$. Moreover, observe that since Sim is a simulator for Π, it follows from the delayed-indistinguishability property that, for all challenges c and $\tilde{c}$ and common inputs x, distribution

$$\{R \leftarrow \{0,1\}^{|x|^d}; (a, \tilde{z}) \leftarrow \mathsf{Sim}(x, \tilde{c}; R); z \leftarrow \mathsf{P}_{\mathsf{sim}}((x, \tilde{c}, R), w, c) : (a, c, z)\}$$

is indistinguishable from

$$\{R \leftarrow \{0,1\}^{|x|^d}; a \leftarrow \mathsf{P}_1(x, w; R); z \leftarrow \mathsf{P}_2(x, w, R, c) : (a, c, z)\}.$$

That is, the two modes of operations of Π are indistinguishable. This property make us able to claim that if Π is WI when a WI challenger interacts with an adversary using $(\mathsf{P}_1, \mathsf{P}_2)$, then Π is WI even when the pair $(\mathsf{Sim}, \mathsf{P}_{\mathsf{sim}})$ is used. Finally, we observe that Chameleon Σ-protocols do exist and Schnorr's protocol [Sch89] is one example. When considering the algorithms associated to a Chameleon Σ-protocol, we will add $\mathsf{P}_{\mathsf{sim}}$.

3.1 Σ-protocols and Witness Indistinguishability

Definition 6. *A* 3*-move protocol* $\Pi = (\mathcal{P}, \mathcal{V})$ *is* Witness Indistinguishable (WI) *for a relation* $\mathcal{R}$ *if, for every malicious verifier* $\mathcal{V}^\star$*, there exists a negligible function* ν *such that for all* x, w, w' *such that* $(x, w) \in \mathcal{R}_L$ *and* $(x, w') \in \mathcal{R}_L$

$$\left|\text{Prob}\left[\ \langle \mathcal{P}(w, 1^\lambda), \mathcal{V}^\star\rangle(x) = 1\ \right] - \text{Prob}\left[\ \langle \mathcal{P}(w', 1^\lambda), \mathcal{V}^\star\rangle(x) = 1\ \right]\right| \le \nu(\lambda).$$

The notion of a *perfect* WI 3-move protocol is obtained by requiring the two distributions to be identical. We start by recalling the following result.

Theorem 3 ([CDS94])**.** *Every Perfect* $\tilde{\Sigma}$*-protocol*[8] *is Perfect WI.*

For completeness, in Appendix A we show a $\tilde{\Sigma}$-protocol that it is not WI.

[8] We remind the reader that we call a 3-move protocol that enjoys Completeness, Special Soundness and Honest-Verifier Zero Knowledge (HVZK) a $\tilde{\Sigma}$-protocol.

3.2 OR Composition of $\tilde{\Sigma}$-protocols: the CDS-OR Transform

In this section we describe the CDS-OR [CDS94] transform in details. Let Π be a $\tilde{\Sigma}$-protocol for polynomial-time relation $\mathcal{R}$ with challenge length l, associated algorithms $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$ and HVZK simulator Sim. The CDS-OR transform constructs a $\tilde{\Sigma}$-protocol Π_{OR} with associated algorithms $(\mathsf{P}_1^{\mathsf{OR}}, \mathsf{P}_2^{\mathsf{OR}}, \mathsf{V}_{\Sigma}^{\mathsf{OR}})$ for the relation

$$\mathcal{R}_{\mathsf{OR}} = \left\{((x_0, x_1), w) : \Big((x_0, w) \in \mathcal{R} \wedge x_1 \in \hat{L}_{\mathcal{R}}\Big) \,\mathsf{OR}\, \Big((x_1, w) \in \mathcal{R} \wedge x_0 \in \hat{L}_{\mathcal{R}}\Big)\right\}.$$

We describe Π_{OR} below.

Protocol 1. *CDS-OR Transform.*
Common input: (x_0, x_1).
$\mathcal{P}$'s private input: (b, w) *with* $b \in \{0, 1\}$ *and* $(x_b, w) \in \mathcal{R}$.

$\mathsf{P}_1^{\mathsf{OR}}((x_0, x_1), (b, w); R_1)$**.** Set $a_b = \mathsf{P}_1(x_b, w; R_1)$. Compute $(a_{1-b}, c_{1-b}, z_{1-b}) \leftarrow \mathsf{Sim}(x_{1-b})$. Output (a_0, a_1).

$P_2^{\mathsf{OR}}((x_0, x_1), (b, w), c, R_1)$**.** Set $c_b = c \oplus c_{1-b}$. Compute $z_b \leftarrow \mathsf{P}_2(x_b, w, c_b, R_1)$. Output $((c_0, c_1), (z_0, z_1))$.

$\mathsf{V}_{\Sigma}^{\mathsf{OR}}((x_0, x_1), (a_0, a_1), c, ((c_0, c_1), (z_0, z_1)))$**.** $\mathsf{V}_{\Sigma}^{\mathsf{OR}}$ accepts if and only if $c = c_0 \oplus c_1$ and $\mathsf{V}(x_0, a_0, c_0, z_0) = 1$ and $\mathsf{V}(x_1, a_1, c_1, z_1) = 1$.

Theorem 4 ([CDS94, GMY06])**.** *If* Π *is a* $\tilde{\Sigma}$*-protocol for* $\mathcal{R}$ *then* Π_{OR} *is a* $\tilde{\Sigma}$*-protocol for* $\mathcal{R}_{\mathsf{OR}}$ *and is WI for relation*

$$\mathcal{R}'_{\mathsf{OR}} = \left\{((x_0, x_1), w) : \Big((x_0, w) \in \mathcal{R} \wedge x_1 \in L_{\mathcal{R}}\Big) \,\mathsf{OR}\, \Big((x_1, w) \in \mathcal{R} \wedge x_0 \in L_{\mathcal{R}}\Big)\right\}.$$

Moreover, if Π *is a Perfect* $\tilde{\Sigma}$*-protocol for* $\mathcal{R}$ *then* Π^{OR} *is WI for* $\mathcal{R}_{\mathsf{OR}}$*.*

It is possible to extend the above construction to handle two different relations $\mathcal{R}_0$ and $\mathcal{R}_1$ that admit $\tilde{\Sigma}$-protocols. Indeed by Theorem 14 (see Appendix A for more details), we can assume, wlog, that $\mathcal{R}_0$ and $\mathcal{R}_1$ have $\tilde{\Sigma}$-protocols Π_0 and Π_1 with the same challenge length. Hence, the construction outlined above can be used to construct $\tilde{\Sigma}$-protocol $\Pi_{\mathsf{OR}}^{\mathcal{R}_0, \mathcal{R}_1}$ for relation

$$\mathcal{R}_{\mathsf{OR}} = \left\{((x_0, x_1), w) : \Big((x_0, w) \in \mathcal{R}_0 \wedge x_1 \in \hat{L}_{\mathcal{R}_1}\Big) \,\mathsf{OR}\, \Big((x_1, w) \in \mathcal{R}_1 \wedge x_0 \in \hat{L}_{\mathcal{R}_0}\Big)\right\}.$$

We have the following theorem.

Theorem 5. *If* Π_0 *and* Π_1 *are* $\tilde{\Sigma}$*-protocols for* $\mathcal{R}_0$ *and* $\mathcal{R}_1$*, respectively, then* $\Pi_{\mathsf{OR}}^{\mathcal{R}_0, \mathcal{R}_1}$ *is a* $\tilde{\Sigma}$*-protocol for relation* $\mathcal{R}_{\mathsf{OR}}$ *and is WI for relation*

$$\mathcal{R}'_{\mathsf{OR}} = \left\{((x_0, x_1), w) : \Big((x_0, w) \in \mathcal{R}_0 \wedge x_1 \in L_{\mathcal{R}_1}\Big) \,\mathsf{OR}\, \Big((x_1, w) \in \mathcal{R}_1 \wedge x_0 \in L_{\mathcal{R}_0}\Big)\right\}.$$

Moreover, if Π_0 *and* Π_1 *are Perfect* $\tilde{\Sigma}$*-protocols for* $\mathcal{R}_0$ *and* $\mathcal{R}_1$ *then* Π^{OR} *is WI for* $\mathcal{R}_{\mathsf{OR}}$*.*

We remark that if Π_0 and Π_1 are Σ-protocols then the CDS-OR transform yields a Σ-protocol for $\mathcal{R}_{\mathsf{OR}}$ and the equivalent of Theorem 5 (and of Theorem 4) holds.

4 t-Instance-Dependent Trapdoor Commitment Schemes

In this section, for integer $t \geq 2$, we define the notion of a t-Instance-Dependent Trapdoor Commitment scheme associated with a polynomial-time relation $\mathcal{R}$ and show constructions for $t = 2$ and $t = 3$.

Definition 7 (t-Instance-Dependent Trapdoor Commitment scheme)**.** *Let $t \geq 2$ be an integer and let $\mathcal{R}$ be a polynomial-time relation. A* t-Instance-Dependent Trapdoor Commitment *(a t-IDTC, in short) scheme for $\mathcal{R}$ with message space M is a triple of PPT algorithms* $(\mathsf{TCom}, \mathsf{TDec}, \mathsf{TFake})$ *where TCom is the randomized* commitment *algorithm that takes as input security parameter 1^λ, an instance $x \in \hat{L}_{\mathcal{R}}$ (with $|x| = \mathtt{poly}(\lambda)$) and a message $m \in M$ and outputs* commitment $\mathtt{com}$, decommitment $\mathtt{dec}$, *and* auxiliary information $\mathtt{rand}$*; TDec is the* verification *algorithm that takes as input $(x, \mathtt{com}, \mathtt{dec}, m)$ and decides whether m is the decommitment of $\mathtt{com}$; TFake is the randomized* equivocation *algorithm that takes as input $(x, w) \in \mathcal{R}$, messages m_1 and m_2 in M, commitment $\mathtt{com}$ of m_1 with respect to instance x and associated auxiliary information $\mathtt{rand}$ and produces decommitment information $\mathtt{dec}_2$ such that TDec, on input $(x, \mathtt{com}, \mathtt{dec}_2, m_2)$, outputs 1.*

A t-Instance-Dependent Trapdoor Commitment scheme has the following properties:

- **Correctness***: for all $x \in \hat{L}_{\mathcal{R}}$, all $m \in M$, it holds that*

$$\text{Prob}\left[\ (\mathtt{com}, \mathtt{dec}, \mathtt{rand}) \leftarrow \mathsf{TCom}(1^\lambda, x, m) : \mathsf{TDec}(x, \mathtt{com}, \mathtt{dec}, m) = 1\ \right] = 1.$$

- t**-Special Extract***: there exists an efficient algorithm* $\mathsf{ExtractTCom}$ *that, on input x, commitment $\mathtt{com}$, pairs $(\mathtt{dec}_i, m_i)_{i=1}^t$ of openings and messages such that*

  - *for $1 \leq i < j \leq t$ we have that $m_i \neq m_j$;*
  - *$\mathsf{TDec}(x, \mathtt{com}, \mathtt{dec}_i, m_i) = 1$, for $i = 1, \ldots, t$;*

  outputs w such that $(x, w) \in \mathcal{R}$.

- **Hiding (resp., Perfect Hiding)***: for every PPT (resp., unbounded) adversary $\mathcal{A}$ there exists a negligible function ν (resp., $\nu(\cdot) = 0$) such that, for all $x \in L_{\mathcal{R}}$ and all $m_0, m_1 \in M$, it holds that*

$$\text{Prob}\left[\ b \leftarrow \{0,1\}; (\mathtt{com}, \mathtt{dec}, \mathtt{rand}) \leftarrow \mathsf{TCom}(1^\lambda, x, m_b) : b = \mathcal{A}(x, \mathtt{com}, m_0, m_1)\ \right] \leq \frac{1}{2} + \nu(\lambda).$$

- **Trapdoorness***: the following two families of probability distributions are indistinguishable:*

$$\{(\mathtt{com}, \mathtt{dec}_1, \mathtt{rand}) \leftarrow \mathsf{TCom}(1^\lambda, x, m_1); \mathtt{dec}_2 \leftarrow \mathsf{TFake}(x, w, m_1, m_2, \mathtt{com}, \mathtt{rand}) : (\mathtt{com}, \mathtt{dec}_2)\}$$

  and

$$\{(\mathtt{com}, \mathtt{dec}_2, \mathtt{rand}) \leftarrow \mathsf{TCom}(1^\lambda, x, m_2) : (\mathtt{com}, \mathtt{dec}_2)\}$$

  over all families $\{(x, w, m_1, m_2)\}$ such that $(x, w) \in \mathcal{R}$ and $m_1, m_2 \in M$.

  The perfect trapdoorness *property requires the two probability distributions to coincide for all (x, w, m_1, m_2) such that $(x, w) \in \mathcal{R}$ and $m_1, m_2 \in M$.*

Constructing a 2-IDTC scheme from a Chameleon Σ-protocol. Let $\Pi = (\mathcal{P}, \mathcal{V})$ with associated algorithms $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{V}, \mathsf{P}_{\mathsf{sim}})$ be a Chameleon Σ-protocol for polynomial-time relation $\mathcal{R}$ with a security parameter 1^λ. Let l be the challenge length of Π and let Sim be a SHVZK simulator associated to Π. We construct a t-IDTC scheme $(\mathsf{TCom}_\Pi, \mathsf{TDec}_\Pi, \mathsf{TFake}_\Pi)$ for $\mathcal{R}$ with messages space $M = \{0,1\}^l$ for $x \in \hat{L}_R$ as follows.

Protocol 2. 2-IDTC scheme from Chameleon Σ-protocol Π.

- $\mathsf{TCom}_\Pi(1^\lambda, x, m_1)$: *On input x and $m_1 \in M$, pick randomness R and compute $(a, z) \leftarrow \mathsf{Sim}(x, m_1; R)$. Output $\mathtt{com} = a$, $\mathtt{dec} = z$ and $\mathtt{rand} = R$;*
- $\mathsf{TDec}_\Pi(x, \mathtt{com}, \mathtt{dec}, m_1)$: *On input $x, \mathtt{com}, \mathtt{dec}$ and m_1, run $b = \mathsf{V}(x, \mathtt{com}, m_1, \mathtt{dec})$ and accept m_1 as the decommitted message iff $b = 1$.*
- TFake_Π: *On input $(x, w) \in \mathcal{R}$, messages $m_1, m_2 \in M$, for m_2 and $\mathtt{rand}$ for $\mathtt{com}$, output $z = \mathsf{P}_{\mathsf{sim}}((x, m_1, \mathtt{rand}), w, m_2)$.*

Theorem 6. *If Π is a Chameleon Σ-protocol for $\mathcal{R}$ then Protocol 2 is a 2-IDTC scheme for $\mathcal{R}$. Moreover, if Π is Perfect then so is Protocol 2.*

Proof. Correctness follows directly from the Completeness property of Π.
2-Special-Extract. Suppose $\mathtt{com}$ is a commitment with respect to instance x and let $\mathtt{dec}_1$ and $\mathtt{dec}_2$ be two openings of $\mathtt{com}$ as messages $m_1 \neq m_2$, respectively. Then, triplets $(\mathtt{com}, m_1, \mathtt{dec}_1)$ and $(\mathtt{com}, m_2, \mathtt{dec}_2)$ are accepting transcripts for Π on common input x with the same first round; that is, they constitute a collision for Π. Therefore, we define algorithm $\mathsf{ExtractTCom}$ to be the algorithm that runs algorithm $\mathsf{Extract}$ (that exists by the special soundness of Π) on input the collision. $\mathsf{ExtractTCom}$ returns the witness for x computed by $\mathsf{Extract}$.
(Perfect) Trapdoorness. It follows from the Perfect Delayed-Indistinguishability property of Π as well as the (perfect) Hiding property.

□

Constructing a 3-IDTC scheme. Let $\mathcal{R}$ be a polynomial-time relation as above admitting a witness-delayed Σ-protocol Π with associated algorithms $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$ and security parameter 1^λ. Let l denote the challenge length of Π. We construct a 3-IDTC scheme for message space $M = \{0,1\}^l$ for $x \in \hat{L}_R$, as follows.

Protocol 3. 3-IDTC scheme.

- TCom_Π**:** *On input 1^λ, x and $m_1 \in M$, pick randomness R and compute $(a_0, z) \leftarrow \mathsf{Sim}(x, m_1)$ and $a_1 \leftarrow \mathsf{P}_1(x; R)$. Let $\mathtt{com}_0 = a_0$ and $\mathtt{com}_1 = a_1$. Output $\mathtt{com} = (\mathtt{com}_b, \mathtt{com}_{1-b})$ for a randomly selected bit b, $\mathtt{dec} = z$ and $\mathtt{rand} = R$.*
- TDec_Π**:** *On input $x, \mathtt{com} = (\mathtt{com}_0, \mathtt{com}_1), \mathtt{dec}$ and m_1, accept m_1 if and only if either $\mathsf{V}(x, \mathtt{com}_0, m_1, \mathtt{dec}) = 1$ or $\mathsf{V}(x, \mathtt{com}_1, m_1, \mathtt{dec}) = 1$.*
- TFake_Π**:** *On input $(x, w) \in \mathcal{R}$, messages $m_1, m_2 \in M$, commitment $\mathtt{com}$ for m_1 and $\mathtt{rand}$ for $\mathtt{com}$, output $z \leftarrow \mathsf{P}_2(x, w, \mathtt{rand}, m_2)$.*

Theorem 7. *If Π is a witness-delayed Σ-protocol for $\mathcal{R}$, with the associated algorithms $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$, then Protocol 3 is a 3-IDTC scheme for $\mathcal{R}$. Moreover, if Π is Perfect then so is Protocol 3.*

Proof. Correctness follows from the completeness of Π.

3-*Special Extract.* It follows from the special soundness of Π. Assume that the committer generates 3 accepting openings $\mathtt{dec}_1$, $\mathtt{dec}_2$ and $\mathtt{dec}_3$, for distinct messages m_1, m_2 and m_3, for the same commitment $\mathtt{com}$ computed w.r.t. x. In this case, we have three accepting transcript for Π and therefore at least two of them must share the same first message, i.e., it is a collision. Thus we can run the extractor $\mathsf{Extract}$ for Π on the collision and obtain a witness for x.

Trapdoorness. It follows from the SHVZK property of Π. We prove this property via hybrid arguments.

The first hybrid, $\mathcal{H}_1$ is the real execution, where a honest prover commits to a message following the honest commitment and decommitment procedure, without using the trapdoor. More formally, in the hybrid $\mathcal{H}_1$ the prover performs the following steps:

- On input x and $m_1, m_2 \in M$, the prover selects random coin tosses R and computes $(a_0, z) \leftarrow \mathsf{Sim}(x, m_2)$, $a_1 \leftarrow \mathsf{P}_1(x; R)$. It picks $b \leftarrow \{0,1\}$ and sends $\mathtt{com} = (a_b, a_{1-b})$, $\mathtt{dec} = z$, m_2.

The second hybrid $\mathcal{H}_2$ is equal to $\mathcal{H}_1$ with the difference that a_0 is computed using the algorithm P_1 and z using P_2. Formally:

- On input x and $m_1, m_2 \in M$, the prover selects random coin tosses $R = (r_1, r_2)$ and computes $a_0 \leftarrow \mathsf{P}_1(x; r_1)$, $z \leftarrow \mathsf{P}_2(x, w, r_1, m_2)$ and $a_1 \leftarrow \mathsf{P}_1(x; r_2)$. It picks $b \leftarrow \{0,1\}$ and sends $\mathtt{com} = (a_b, a_{1-b})$, $\mathtt{dec} = z$, m_2.

Due to the SHVZK property of Π, $\mathcal{H}_1$ is indistinguishable from $\mathcal{H}_2$. Now we consider the hybrid $\mathcal{H}_3$ in which a_1 is computed using $\mathsf{Sim}(x, m_2)$. Formally:

- On input x and $m_1, m_2 \in M$, the prover selects random coin tosses R and computes $a_0 \leftarrow \mathsf{P}_1(x; R)$, $z \leftarrow \mathsf{P}_2(x, w, R, m_2)$ and $(a_1, \overline{z}) \leftarrow \mathsf{Sim}(x, m_1)$. It picks $b \leftarrow \{0,1\}$ and sends $\mathtt{com} = (a_b, a_{1-b})$, $\mathtt{dec} = z$, m_2.

Even in this case, we can claim that $\mathcal{H}_3$ is indistinguishable from $\mathcal{H}_2$ because of the SHVZK of Π. The proof ends with the observation that $\mathcal{H}_3$ is the experiment in which a sender commits to a message m_1 and opens to m_2 using the trapdoor.

If Π is a perfect SHVZK protocol, then the sequence of hybrids produces identical distributions.

□

5 Our New OR-Composition Technique

In this section we formally describe our new OR transform. Let $\mathcal{R}_0$ be a relation admitting a t-IDTC scheme, $I = (\mathsf{TCom}_{\Pi_0}, \mathsf{TDec}_{\Pi_0}, \mathsf{TFake}_{\Pi_0})$, with $t = 2$ or $t = 3$, and $\mathcal{R}_1$ a relation admitting an input-delayed Σ-protocol Π_1 with associated algorithms $(\mathsf{P}_1^1, \mathsf{P}_2^1, \mathsf{V}^1)$ and simulator Sim^1. We show a Σ-protocol Π^{OR} for the OR relation:

$$\mathcal{R}_{\mathsf{OR}} = \{((x_0, x_1), w) : ((x_0, w) \in \mathcal{R}_0 \wedge x_1 \in \hat{L}_{\mathcal{R}_1}) \text{ OR } ((x_1, w) \in \mathcal{R}_1 \wedge x_0 \in \hat{L}_{\mathcal{R}_0})\}.$$

We denote by $(\mathsf{P}_1^{\mathsf{OR}}, \mathsf{P}_2^{\mathsf{OR}}, \mathsf{V}^{\mathsf{OR}})$ the algorithms associated with Π^{OR}. We assume that the initial common input is x_0. The other input x_1 and the witness w for (x_0, x_1) are made available to the prover only after the challenge has been received. We let $b \in \{0,1\}$ be such that $(x_b, w) \in \mathcal{R}_b$ and assume that the message space of the t-IDTC scheme I includes all possible first-round messages of

Π_1. Note that for the constructions of the t-IDTC scheme we provide, the message space coincides with the set of challenges of the underlying Σ-protocol and, in Appendix A.1, we show that the challenge length of a Σ-protocol can be easily expanded/reduced.

We remind that prover algorithm $\mathsf{P}_2^{\mathsf{OR}}$ receives as further input the randomness $(R_1, \mathtt{rand}_1)$ used by $\mathsf{P}_1^{\mathsf{OR}}$ to produce the first-round message.

Protocol 4. Protocol Π^{OR} for $\mathcal{R}_{\mathsf{OR}}$.

Common input: $(x_0, 1^\lambda)$, *where* 1^λ *is the security parameter.*

1. $\mathsf{P}_1^{\mathsf{OR}}(x_0, 1^\lambda)$. *Pick random* R_1 *and compute* $a_1 \leftarrow \mathsf{P}_1^1(1^\lambda; R_1)$. *Then commit to* a_1 *by running* $(\mathtt{com}, \mathtt{dec}_1, \mathtt{rand}_1) \leftarrow \mathsf{TCom}_{\Pi_0}(1^\lambda, x_0, a_1)$. *Output* $\mathtt{com}$.
2. $\mathsf{P}_2^{\mathsf{OR}}((x_0, x_1), c, (w, b), (\mathtt{rand}_1, R_1))$ *(with* $(x_b, w) \in \mathcal{R}_b$*).*

   If $b = 1$, *compute* $z_1 \leftarrow \mathsf{P}_2^1(x_1, w, R_1, c)$ *and output* $(\mathtt{dec}_1, a_1, z_1)$.

   If $b = 0$, *compute* $(a_2, z_2) \leftarrow \mathsf{Sim}^1(x_1, c)$, $\mathtt{dec}_2 \leftarrow \mathsf{TFake}_{\Pi_0}(x_0, w, a_1, a_2, \mathtt{com}, \mathtt{rand}_1)$ *and output* $(\mathtt{dec}_2, a_2, z_2)$.
3. V^{OR}, *on input* (x_0, x_1), $\mathtt{com}$, c, *and* $(\mathtt{dec}, a, z))$ *received from* Π^{OR}, *outputs* 1 *iff*

$$\mathsf{TDec}_{\Pi_0}(x_0, \mathtt{com}, \mathtt{dec}, a) = 1 \text{ and } \mathsf{V}^1(x_1, a, c, z) = 1;$$

Theorem 8. *If* $\mathcal{R}_0$ *admits a* 2-IDTC *(resp.,* 3-IDTC*) scheme and if* $\mathcal{R}_1$ *admits an input-delayed* Σ*-protocol, then* Π^{OR} *is a* Σ*-protocol (resp., is a 3-round public-coin SHVZK PoK) for relation* $\mathcal{R}_{\mathsf{OR}}$.

Proof. Completeness follows by inspection. We next prove the properties of Protocol 4 when instantiated with a 2-IDTC and 3-IDTC schemes.

Proof for the construction based on the 2-IDTC scheme.

Special Soundness. It follows from the special soundness of the underlying Σ-protocol Π_1 and the 2-Special Extract of the 2-IDTC scheme. More formally, consider a collision $(\mathtt{com}, c, (\mathtt{dec}, a, z))$ and $(\mathtt{com}, c', (\mathtt{dec}', a', z'))$ for input (x_0, x_1). We observe that:

- if $a = a'$ then (a, c, z) and (a', c', z') is a collision for Π_1 for input x_1; then we can obtain a witness w_1 for x_1 by the Special Soundness property of Π_1;
- if $a \neq a'$, then $\mathtt{dec}$ and $\mathtt{dec}'$ are two openings of $\mathtt{com}$ with respect to x_0 for messages $a \neq a'$; then we can obtain a witness w_0 by the 2-Special Extract of the 2-IDTC scheme.

SHVZK property. Consider simulator $\mathsf{Sim}^{\mathsf{OR}}$ that, on input (x_0, x_1) and challenge c, sets $(a, c, z) \leftarrow \mathsf{Sim}_1(x_1, c)$ and $(\mathtt{com}, \mathtt{dec}) \leftarrow \mathsf{TCom}_{x_0}(a)$, and outputs $(\mathtt{com}, c, (\mathtt{dec}, a, z))$. Next, we show that the transcript generated by $\mathsf{Sim}^{\mathsf{OR}}$ is indistinguishable from the one generated by a honest prover.

Let us first consider the case in which the prover of Π^{OR} receives a witness for x_1. In this case, if we sample a random distribution $(\mathtt{com}, c, (\mathtt{dec}, a, z))$ of Π^{OR} on input (x_0, x_1) constrained to c being the challenge we have that (a, c, z) has the same distribution as in random transcript of Π_1 on input x_1 constrained to c being the challenge; moreover, $(\mathtt{com}, \mathtt{dec})$ is a pair of commitment and decommitment of a with respect to x_0. By the property of Sim_1, this distribution is indistinguishable from (a, c, z) computed as $\mathsf{Sim}_1(x_1, c)$ which is exactly as in the output $\mathsf{Sim}^{\mathsf{OR}}$.

Let us now consider the case in which the prover of Π^{OR} receives a witness for x_0. If we sample a random distribution $(\mathtt{com}, c, (\mathtt{dec}, a, z))$ of Π^{OR} on input (x_0, x_1) constrained to c being the challenge we have that (a, c, z) are distributed exactly as in the output of $\mathsf{Sim}^{\mathsf{OR}}$ (that is by running Sim_1 on input x_1 and c). In addition, in the output of $\mathsf{Sim}^{\mathsf{OR}}$, $(\mathtt{com}, \mathtt{dec})$ are commitment and decommitment of a whereas in the view of Π^{OR} they are computed by means of TFake algorithm. However, the two distributions are indistinguishable by the trapdoorness of the Instance-Dependent Trapdoor Commitment.

Proof for the construction based on the 3-IDTC scheme.

3-*Special Soundness.* This property ensures that there exists an efficient algorithm that, given three accepting transcripts, (a, c_0, z_0), (a, c_1, z_1), (a, c_2, z_2) with $c_i \neq c_j$ for $1 \leq i < j \leq 3$, for the same common input, outputs a witness for x.

Consider three accepting transcripts for Π^{OR} and input (x_0, x_1):

$$(\mathtt{com}, c_1, (\mathtt{dec}_1, a_1, z_1)), \qquad (\mathtt{com}, c_2, (\mathtt{dec}_2, a_2, z_2))$$

and

$$(\mathtt{com}, c_3, (\mathtt{dec}_3, a_3, z_3)).$$

We observe that:

- if $a_i = a_j$ for some $i \neq j$ then (a_i, c_i, z_i) and (a_j, c_j, z_j) is a collision for Π_1 for input x_1; thus we can obtain a witness w_1 for x_1 by the Special Soundness property of Π_1;

- if $a_i \neq a_j$ for all $i \neq j$, then, $\mathtt{dec}_1$ and $\mathtt{dec}_2$ and $\mathtt{dec}_3$ are three openings of the same $\mathtt{com}$ with respect to x_0 for messages a_1, a_2 and a_3; then we can obtain a witness w_0 for x_0 by the 3-Special Extract of the 3-IDTC scheme.

We stress that having a long enough challenge, 3-special soundness implies the proof of knowledge property.

SHVZK property. This is similar to the proof for the construction based on 2-IDTC. □

5.1 Witness Indistinguishability of Our Transform

In this section we discuss the *adaptive WI* property of Π^{OR}. Roughly speaking, adaptive WI means that in the WI experiment the adversary $\mathcal{A}$ is not forced to choose *both* theorems x_0 and x_1 at the onset of the experiment. Rather, she can choose theorem x_1 and witnesses w_0, w_1 adaptively, *after* seeing the first message of Π^{OR} played by the prover on input x_0. After x_1, w_0, w_1 have been selected by $\mathcal{A}$, the experiment randomly selects $b \leftarrow \{0, 1\}$. The prover then receives x_1 and w_b and proceeds to complete the protocol. The adversary wins the game if she can guess b with probability non-negligibly greater than 1/2. More formally, we consider adaptive WI for polynomial-time relation

$$\mathcal{R}^p_{\mathsf{OR}} = \Big\{((x_0, x_1), w) : \Big((x_0, w) \in \mathcal{R}_0 \wedge x_1 \in \hat{L}_{\mathcal{R}_1}\Big) \,\mathsf{OR}\, \Big((x_1, w) \in \mathcal{R}_1 \wedge x_0 \in \hat{L}_{\mathcal{R}_0}\Big)\Big\}$$

and for the weaker relation

$$\mathcal{R}^c_{\mathsf{OR}} = \Big\{((x_0, x_1), w) : \Big((x_0, w) \in \mathcal{R}_0 \wedge x_1 \in L_{\mathcal{R}_1}\Big) \;\mathsf{OR}\; \Big((x_1, w) \in \mathcal{R}_1 \wedge x_0 \in L_{\mathcal{R}_0}\Big)\Big\}.$$

The adaptive WI experiment, $\mathsf{ExpWI}^{\delta}_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$ with $\delta \in \{c, p\}$, is parameterized by PPT adversary $\mathcal{A}$ and has three inputs: instance x_0, security parameter λ, and auxiliary information aux for $\mathcal{A}$.

$\mathsf{ExpWI}^{\delta}_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$:

1. $a = \mathsf{P}^{\mathsf{OR}}_1(x_0, 1^\lambda; R_1)$, for random coin tosses R_1;
2. $\mathcal{A}(x_0, a, \mathsf{aux})$ outputs $((x_1, w_0, w_1), c, \mathtt{state})$ such that $((x_0, x_1), w_0), ((x_0, x_1), w_1) \in \mathcal{R}^{\delta}_{\mathsf{OR}}$;
3. $b \leftarrow \{0, 1\}$;
4. $z \leftarrow \mathsf{P}^{\mathsf{OR}}_2((x_0, x_1), w_b, R_1, c)$;
5. $b' \leftarrow \mathcal{A}(z, \mathtt{state})$;
6. If $b = b'$ then output 1 else output 0.

We set
$$\mathsf{Adv}^{\delta}_{\mathcal{A}}(x_0, \lambda, \mathsf{aux}) = \left|\text{Prob}\left[\ \mathsf{ExpWI}^{\delta}_{\mathcal{A}}(x_0, \lambda, \mathsf{aux}) = 1\ \right] - \frac{1}{2}\right|$$

Definition 8. Π^{OR} *is* Adaptive Witness Indistinguishable *(resp.,* Adaptive Perfect Witness Indistinguishable*) if for every adversary $\mathcal{A}$ there exists a negligible function ν such that for all* aux *and x_0 it holds that $\mathsf{Adv}^{c}_{\mathcal{A}}(x_0, \lambda, \mathsf{aux}) \leq \nu(\lambda)$ (resp., $\mathsf{Adv}^{p}_{\mathcal{A}}(x_0, \lambda, \mathsf{aux}) = 0$).*

Next, in Theorem 9, we prove the Adaptive Perfect WI of Π^{OR} when both Π_0 and Π_1 are perfect SHVZK. When one of Π_0 and Π_1 is not perfect, we would like to prove that Π^{OR} is Adaptive WI. In Theorem 10 we prove a weaker form of Adaptive WI in which the adversary is restricted in his choice of witnesses (w_0, w_1) for relation $\mathcal{R}^{c}_{\mathsf{OR}}$. We leave open the problem of an OR-composition technique that gives Adaptive WI when the Σ-protocol composed are not both perfect SHVZK.

Theorem 9. *If Π_0 and Π_1 are perfect SHVZK then Π^{OR} is Adaptive Perfect Witness Indistinguishable.*

Proof. The proof considers the following three cases:

Case 1. $(x_0, w_0) \in \mathcal{R}_0$ and $(x_1, w_1) \in \mathcal{R}_1$;

Case 2. $(x_0, w_0) \in \mathcal{R}_0$ and $(x_0, w_1) \in \mathcal{R}_0$;

Case 3. $(x_1, w_0) \in \mathcal{R}_1$ and $(x_1, w_1) \in \mathcal{R}_1$.

For each case we present a sequence of hybrids and prove that pairs of consecutive hybrids are perfectly indistinguishable.

Case 1. The first hybrid experiment $\mathcal{H}_1(x_0, \lambda, \mathsf{aux})$ is the original experiment $\mathsf{ExpWI}^{p}_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$ in which $b = 1$ (and thus $\mathcal{P}$ uses witness w_1). That is,

- In Step 1 of $\mathsf{ExpWI}^{p}_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$, the following steps are executed:
    1. $a = \mathsf{P}^{1}_1(1^\lambda; R_1)$, for random coin tosses R_1;

2. $(\mathtt{com}, \mathtt{dec}, \mathtt{rand}) \leftarrow \mathsf{TCom}_{\Pi_0}(x_0, 1^\lambda, a)$ and outputs $\mathtt{com}$.

- In Step 4 of $\mathsf{ExpWI}^p_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$, the following steps are executed:
  1. set $a' = a$;
  2. $z \leftarrow \mathsf{P}^1_2(x_1, w_1, c, R_1)$;
  3. set $\mathtt{dec}' = \mathtt{dec}$;
  4. output $(\mathtt{dec}', a', z)$.

The second hybrid experiment $\mathcal{H}_2(x_0, \lambda, \mathsf{aux})$ differs from $\mathcal{H}_1(x_0, \lambda, \mathsf{aux})$ in the way a' and $\mathtt{dec}'$ are computed. More specifically,

- Step 1 of $\mathsf{ExpWI}^p_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$ stays the same.
  1. $a = \mathsf{P}^1_1(1^\lambda; R_1)$, for random coin tosses R_1;
  2. $(\mathtt{com}, \mathtt{dec}, \mathtt{rand}) \leftarrow \mathsf{TCom}_{\Pi_0}(x_0, 1^\lambda, a)$ and outputs $\mathtt{com}$.
- In Step 4 of $\mathsf{ExpWI}^p_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$, the following steps are executed:
  1. $a' = \mathsf{P}^1_1(1^\lambda; R'_1)$, for random coin tosses R'_1;
  2. $z \leftarrow \mathsf{P}^1_2(x_1, w_1, c, R'_1)$;
  3. $\mathtt{dec}' \leftarrow \mathsf{TFake}_{\Pi_0}(x_0, w_0, a, a', \mathtt{com}, \mathtt{rand})$;
  4. $(\mathtt{dec}', a', z)$.

The trapdoorness of the instance-dependent trapdoor commitment scheme based on Π_0 guarantees that $\mathcal{H}_1(x_0, \lambda, \mathsf{aux})$ and $\mathcal{H}_2(x_0, \lambda, \mathsf{aux})$ are perfectly indistinguishable for all λ.

The third hybrid experiment $\mathcal{H}_3(x_0, \lambda, \mathsf{aux})$ differs from $\mathcal{H}_2(x_0, \lambda, \mathsf{aux})$ in the way a' and z are computed. More specifically,

- Step 1 of $\mathsf{ExpWI}^p_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$ stays the same.
  1. $a = \mathsf{P}^1_1(1^\lambda; R_1)$, for random coin tosses R_1;
  2. $(\mathtt{com}, \mathtt{dec}, \mathtt{rand}) \leftarrow \mathsf{TCom}_{\Pi_0}(x_0, 1^\lambda, a)$ and outputs $\mathtt{com}$.
- In Step 4 of $\mathsf{ExpWI}^p_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$, the following steps are executed:
  1. $(a', z) \leftarrow \mathsf{Sim}^1(x_1, c)$;
  2. $\mathtt{dec}' \leftarrow \mathsf{TFake}_{\Pi_0}(x_0, w_0, a, a', \mathtt{com}, \mathtt{rand})$;
  3. $(\mathtt{dec}', a', z)$.

By the perfect SHVZK of Π_1, we have that $\mathcal{H}_2(x_0, \lambda, \mathsf{aux})$ and $\mathcal{H}_3(x_0, \lambda, \mathsf{aux})$ are perfectly indistinguishable for all λ. The proof ends with the observation that $\mathcal{H}_3(x_0, \lambda, \mathsf{aux})$ is exactly experiment $\mathsf{ExpWI}^p_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$ when $b = 0$.

Case 2. The first hybrid experiment $\mathcal{H}_1(x_0, \lambda, \mathsf{aux})$ is again the original experiment $\mathsf{ExpWI}^{p}_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$ in which $b = 1$ (and thus $\mathcal{P}$ uses witness w_1). The second hybrid experiment $\mathcal{H}_2(x_0, \lambda, \mathsf{aux})$ differs from $\mathcal{H}_1(x_0, \lambda, \mathsf{aux})$ in the way TFake is executed (namely, using as input w_0 instead of w_1). More specifically,

- Step 1 of $\mathsf{ExpWI}^{p}_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$ stays the same.
  1. $a = \mathsf{P}^1_1(1^\lambda; R_1)$, for random coin tosses R_1;
  2. $(\mathtt{com}, \mathtt{dec}, \mathtt{rand}) \leftarrow \mathsf{TCom}_{\Pi_0}(x_0, 1^\lambda, a)$ and outputs $\mathtt{com}$.
- In Step 4 of $\mathsf{ExpWI}^{p}_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$, the following steps are executed:
  1. $(a', z) = \mathsf{Sim}^1(x_1, c)$;
  2. $\mathtt{dec}' \leftarrow \mathsf{TFake}_{\Pi_0}(x_0, w_0, a, a', \mathtt{com}, \mathtt{rand})$;
  3. $(\mathtt{dec}', a', z)$.

The trapdoorness of the instance-dependent trapdoor commitment scheme based on Π_0 implies that $\mathcal{H}_1(x_0, \lambda\mathsf{aux})$ is perfectly indistinguishable from $\mathcal{H}_2(x_0, \lambda\mathsf{aux})$ for all λ. The proof ends with the observation that $\mathcal{H}_2(x_0, \lambda, \mathsf{aux})$ is exactly experiment $\mathsf{ExpWI}^{p}_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$ when $b = 0$.

Case 3. The first hybrid experiment $\mathcal{H}_1(x_0, \lambda, \mathsf{aux})$ is again the original experiment $\mathsf{ExpWI}^{p}_{\mathcal{A}}(x_0, \mathsf{aux})$ in which $b = 1$ (and thus $\mathcal{P}$ uses witness w_1). The second hybrid experiment $\mathcal{H}_2(x_0, \lambda, \mathsf{aux})$ differs from $\mathcal{H}_1(x_0, \lambda, \mathsf{aux})$ in the way z is computed (using as input w_1 instead of w_0 when P_2 is executed). More specifically,

- In Step 1 of $\mathsf{ExpWI}^{p}_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$, the following steps are executed:
  1. $a = \mathsf{P}^1_1(1^\lambda; R_1)$, for random coin tosses R_1;
  2. $(\mathtt{com}, \mathtt{dec}, \mathtt{rand}) \leftarrow \mathsf{TCom}_{\Pi_0}(x_0, 1^\lambda, a)$ and outputs $\mathtt{com}$.
- In Step 4 of $\mathsf{ExpWI}^{p}_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$, the following steps are executed:
  1. $z \leftarrow \mathsf{P}^1_2(x_1, w_0, c, R_1)$;
  2. output $(\mathtt{dec}, a, z)$

The Perfect WI property of Π_1 implies that $\mathcal{H}_1(x_0, \lambda, \mathsf{aux})$ is perfectly indistinguishable from $\mathcal{H}_2(x_0, \lambda, \mathsf{aux})$. The proof ends with the observation that $\mathcal{H}_2(x_0, \lambda, \mathsf{aux})$ is exactly the experiment $\mathsf{ExpWI}^{p}_{\mathcal{A}}(x_0, \lambda, \mathsf{aux})$ when $b = 0$. □

Next we consider the computational case in which one of the two components is not Perfect SHVZK.

Theorem 10. *If Π_0 and Π_1 are SHVZK then Π^{OR} is Adaptive Witness Indistinguishable with respect to adversaries that output (x_1, w_0, w_1) such that at least one of w_0 and w_1 is a witness for $x_1 \in L_{\mathcal{R}_1}$.*

Proof. We prove this theorem by considering the following two cases:

1. $(x_0, w_0) \in \mathcal{R}_0$ and $(x_1, w_1) \in \mathcal{R}_1$;
2. $(x_1, w_0) \in \mathcal{R}_1$ and $(x_1, w_1) \in \mathcal{R}_1$.

Case 1. In this case the proof follows closely the one of Case 1 of Theorem 9, with the difference that hybrids here are only computationally indistinguishable.

Case 2. In this case we show that there exists $\mathcal{A}'$ for Case 1 that has the same success probability of $\mathcal{A}$. Suppose indeed that both w_0 and w_1 are witnesses for x_1 and that $\mathcal{A}$ breaks the adaptive WI property of Π^{OR}. Then, by definition of $\mathcal{R}^c_{\mathsf{OR}}$ and by Definition 8, there exists $\mathcal{A}'$ that has in his description a witness w_2 for x_0. Indeed, the output of $\mathcal{A}$ interacting with $\mathcal{P}((x_0, x_1), w_2)$ would necessarily be distinguishable from the output of the interaction with either $\mathcal{P}((x_0, x_1), w_0)$ or $\mathcal{P}((x_0, x_1), w_1)$. Therefore $\mathcal{A}'$ would contradict Case 1 and thus there exists no successful $\mathcal{A}$ for Case 2. $\square$

6 Applications

In this section, we describe the application of our new OR-composition technique for constructing a 3-round straight-line perfect quasi-polynomial time simulatable argument system, an efficient 4-round resettable WI argument system and an efficient 4-round resettable zero knowledge with concurrent soundness argument system in the BPK model. For the last application we provide only an informal protocol description.

6.1 A 3-Round Efficient Perfectly Simulatable Argument System

In [Pas03], Pass introduced relaxed notions of zero knowledge and knowledge extraction in which the simulator and the extractor are allowed to run in quasi-polynomial time. Allowing the simulator to run in quasi-polynomial time typically dispenses with the need of rewinding the verifier; that is, the simulator is *straight-line*. In [Pas03], Pass first describes the following 2-round perfect ZK argument for any language L: the verifier $\mathcal{V}$ sends a value $Y = f(y)$ for a randomly chosen y where f is a sub-exponentially hard OWF and the first round of a ZAP protocol. The prover $\mathcal{P}$ then sends a commitment to $(y'|w')$ and uses the second round of the ZAP to prove that either $y' = f^{-1}(y)$ or w' is a witness for $x \in L$. If language L admits a Σ-protocol Π_L then the above construction can be implemented as an efficient 4-round argument with quasi-polynomial time simulation: the function f is concretely instantiated to be an exponentiation in a group in which the Discrete Log problem is hard and the ZAP is replaced with the CDS-OR composition of Π_L and Schnorr's Σ-protocol for the Discrete Log.

Note that Schnorr's Σ-protocol is input delayed and thus we can use it as Σ-protocol Π_1 in our OR transform in conjunction with any Chameleon Σ-protocol Π_0. One drawback of reducing to three round the result of [Pas03] is that we can use only a perfect Σ-protocolsince our goal is to obtain perfect WI in just three rounds.

6.1.1 Preliminary Definitions

We start by providing some useful definitions.

Simulation in quasi-polynomial time. Since the verifier in an interactive argument is often modeled as a PPT machine, the classical zero-knowledge definition requires that the simulator runs also in (expected) polynomial time. In [Pas03], the simulator is allowed to run in time $\lambda^{\mathtt{poly}(\mathtt{log}(\lambda))}$. Loosely speaking, we say that an interactive argument is $\lambda^{\mathtt{poly}(\mathtt{log}(\lambda))}$-perfectly simulatable if for any adversarial verifier there exists a simulator running in time $\lambda^{\mathtt{poly}(\mathtt{log}(\lambda))}$, where λ is the size of the statement being proved, whose output is identically distributed to the output of the adversarial verifier.

Definition 9 (One-way functions for sub-exponential circuits. [Pas03])**.** *A function $f : \{0,1\}^* \to \{0,1\}^*$ is called one-way for sub-exponential circuits if there exists a constant α such that the following two condition holds:*

- *there exist a deterministic polynomial-time algorithm that on input y outputs $f(y)$;*
- *for every probabilistic algorithm $\mathcal{A}$ with running time bounded by 2^{λ^α}, all sufficiently large λ's, and every auxiliary input $z \in \{0,1\}^{\mathtt{poly}(\lambda)}$*

$$\text{Prob}\left[\ y \stackrel{R}{\leftarrow} \{0,1\}^* : \mathcal{A}(f(y), z) \in f^{-1}(f(y))\ \right] < \frac{1}{\mathtt{poly}(2^{\lambda^\alpha})}.$$

Now we define straight-line $T(\lambda)$-perfectly simulatable interactive arguments.

For our result we consider a one-way functions for sub-exponential circuits that is also one-to-one.

Definition 10 (straight-line $T(\lambda)$ simulatability, Def. 31 of [Pas04])**.** *Let $T(\lambda)$ be a class of functions that is closed under composition with any polynomial. We say that an interactive argument (proof) $(\mathcal{P}, \mathcal{V})$ for the language $L \in NP$, with the witness relation $\mathcal{R}_L$, is* straight-line *$T(\lambda)$-simulatable if for every PPT machine $\mathcal{V}^\star$ there exists a probabilistic simulator S with running time bounded by $T(\lambda)$ such that the following two ensembles are computationally indistinguishable (when the distinguish gap is a function in $\lambda = |x|$)*

- $\{(\langle \mathcal{P}(w), \mathcal{V}^\star(z)\rangle(x))\}_{z\in\{0,1\}^*, x\in L}$ *for arbitrary w s.t. $(x, w) \in \mathcal{R}_L$*
- $\{(\langle S, \mathcal{V}^\star(z)\rangle(x))\}_{z\in\{0,1\}^*, x\in L}$

We note that the above definition is very restrictive. In fact, the simulator is supposed to act as a cheating prover, with its only advantage being the possibility of running in time $T(\lambda)$, instead of in polynomial time. Trivially, it do not exist a straight-line $T(\lambda)$-simulatable proof for non-trivial languages (this should be contrasted with straight-line simulatable interactive arguments, which instead do exist).

The following theorem shows the importance of straight-line $\lambda^{\mathtt{poly}(\mathtt{log}(\lambda))}$-perfect simulatability by connecting it to concurrent composition of arguments.

Theorem 11. *If the interactive argument $\Pi = (\mathcal{P}, \mathcal{V})$ is straight-line $\lambda^{\mathtt{poly}(\mathtt{log}(\lambda))}$-simulatable then it is also straight-line concurrent $\lambda^{\mathtt{poly}(\mathtt{log}(\lambda))}$-simulatable.*

6.1.2 The Protocol

For any NP-language L we consider the perfect chameleon Σ-protocol Π_L for the relation R_L. Also we consider the Schnorr Σ-protocol Π_{DLOG} the following relation

$$\mathsf{DLOG} = \{((\mathcal{G}, q, g, Y), y) : g^y = Y\}$$

with the associated NP-language L_{DLOG}, over groups $\mathcal{G}$ of prime-order q, and use our OR-composition technique to obtain a new perfect Σ-protocol $\Pi^{\mathsf{OR}} = (\mathcal{P}^{\mathsf{OR}}, \mathcal{V}^{\mathsf{OR}})$ for the relation

$$\mathcal{R}_{\mathsf{OR}} = \left\{((x_L, x_{\mathsf{DLOG}}), w) : \left((x_L, w) \in \mathcal{R}_L \wedge x_{\mathsf{DLOG}} \in \hat{L}_{\mathsf{DLOG}}\right) \mathsf{OR} \left((x_{\mathsf{DLOG}}, w) \in \mathsf{DLOG} \wedge x_L \in \hat{L}_{\mathcal{R}_L}\right)\right\}$$

with challenge length $l = \lambda$ and associated algorithms $\mathsf{P}_1^{\mathsf{OR}}$, $\mathsf{P}_2^{\mathsf{OR}}$ and V^{OR}.

Let f be a sub-exponentially hard one-to-one one-way function implemented using DLog as described before, with the only change that for some constant α, f is one-way w.r.t circuits of size 2^{λ^α}. Let $L \in$NP and $k = \frac{1}{\alpha} + 1$. Our 3-round straight-line quasi-polynomial time simulatable argument system for $x \in L$ is the following.

Protocol 5. A 3-round straight-line quasi-polynomial time simulatable argument system.
Common input: *An instance x of a language $L \in NP$ with witness relation R_L and a perfect chameleon Σ-protocol, and 1^λ as security parameter.*
Private input: *$\mathcal{P}$ has w as a private input, s.t. $(x, w) \in \mathcal{R}_L$.*
Round 1. *$\mathcal{P} \to \mathcal{V}$:*

1. *On input a randomness R_1, $\mathcal{P}$ uniformly chooses (p, q, g) where $p = 2q + 1$ is a safe prime and g is a generator of a group $\mathcal{G}_q$ of size q. We remark that (p, q, g) are parameters selected so that the function $f(y) = g^y$ is a one-to-one one-way function for some constant α w.r.t circuits of size 2^{λ^α}.*
2. *$\mathcal{P}$ computes $a \leftarrow \mathsf{P}_1^{\mathsf{OR}}((x, 1^{\lambda^\alpha}); R_1)$.*
3. *$\mathcal{P}$ sends (p, q, g) and a to $\mathcal{V}$.*

Round 2. *$\mathcal{V} \to \mathcal{P}$:*

1. *$\mathcal{V}$ chooses $y \leftarrow \mathbb{Z}_q$ and computes $Y = g^y$.*
2. *$\mathcal{V}$ chooses $c \leftarrow \{0, 1\}^l$.*
3. *$\mathcal{V}$ sends c and Y to $\mathcal{P}$.*

Round 3. *$\mathcal{P} \to \mathcal{V}$:*

1. *$\mathcal{P}$ computes $z \leftarrow \mathsf{P}_2^{\mathsf{OR}}((x, ((p, q, g), Y)), w, c, R_1)$.*
2. *$\mathcal{P}$ sends z to $\mathcal{V}$.*
3. *$\mathcal{V}$ accepts if and only if $\mathsf{V}^{\mathsf{OR}}((x, ((p, q, g), Y)), a, c, z) = 1$.*

We remark that we are using the same assumption of [CD08] that allows the adversary of DLog to generate the DLog parameters while the challenger selects the random element of the group.

Theorem 12. *If Π^{OR} is a perfect Σ-protocol for OR composition of $\mathcal{R}_L$ and DLOG, then Protocol 5 is a 3-round straight-line perfectly $\lambda^{O(\log^k \lambda)}$-simulatable argument of knowledge.*

Proof. Completeness follows directly from the completeness of Π^{OR}.

Soundness/knowledge extraction. We show that Π is an argument of knowledge; this directly implies soundness. The claim follows from the fact that the argument system Π^{OR} used is a proof of knowledge when the challenge is long enough. and from the fact that a PPT adversary only finds a pre-image to Y (for f) with negligible probability. More formally, we construct a polynomial-time extractor E for every polynomial-time $\mathcal{P}^\star$ for protocol Π. E internally incorporates $\mathcal{P}^\star$ and each time Π^{OR} proves a new theorem it proceeds as follows. E invokes the extractor E^{OR} for Π^{OR}. E outputs whatever E^{OR} outputs. By the proof knowledge property of Π^{OR}, the output of E will either be a witness w for the statement proved, or the pre-image of Y. If E outputs w, we are done. Otherwise, if it outputs y with non-negligible probability, then we can construct a reduction that breaks the DLog assumption (still in the form proposed by [CD08]).

Quasi-polynomial time perfect simulation. Consider a straight-line simulator Sim that computes the first round as the honest prover. This is possible because Π^{OR} does not need any witness to computes the first round. After the simulator receives Y it checks that Y has a pre-image. Sim thereafter performs an exhaustive search to find a pre-image y of a value Y for the function f. To perform this task Sim tries all possible values $y' \in \{0,1\}^{\log^k \lambda}$ and checks if $f(y') = Y$. This thus takes time $\mathtt{poly}(2^{\log^k \lambda})$, since the time it takes to evaluate the function f is a polynomial in λ. After having found a value y such that $f(y) = Y$, Sim uses y as witness to complete the execution of Π^{OR} (instead of using a real witness for x, as the honest prover would do). Clearly the running time of Sim is bounded by $\lambda^{O(\log^k \lambda)}$. We proceed to show that the output of the simulator is identically distributed to the output of any adversarial verifier in a real execution with an honest prover. Note that the only difference between a real execution and a simulated execution is in the choice of the witness used in the last stage of the protocol. Therefore, from the adaptive WI property of Π^{OR} we have that the output of the simulated execution is identically distributed to the output of the real execution. $\square$

6.2 Efficient Resettable WI Argument System

In this section, we show how to efficiently transform *any* Σ-protocol Π into a resettable WI (rWI) argument system at the cost of adding only one extra round and a constant number of modular exponentiations.

Resettable witness indistinguishability was introduced in [CGGM00]. Very roughly, a resetting verifier is a PPT adversary that is able to interact with the prover polynomially many times with possibly distinct inputs forcing the prover to execute the protocol using the same randomness several times. Namely, we can think of a prover under a reset attack as being equipped with a vector of inputs $\bar{x}$ and a vector of random tapes $\vec{\omega}$. The malicious verifier can adaptively start a new interaction with the prover by specifying the input x_i and the randomness ω_j to be used in the interaction. Moreover, the malicious verifier has complete control over the schedule of the messages. A concurrent verifier is a restricted form of resetting verifier that cannot start two interactions with the same randomness. A formal definition is found in [CGGM00].

An informal description. Let $\mathcal{R}$ be a polynomial-time relation with Σ-protocol Π. We describe our 4-round rWI argument system $\Pi^{\mathsf{WI}} = (\mathcal{P}, \mathcal{V})$ for $\mathcal{R}$ incrementally. The basic idea is to have $\mathcal{V}$ commit to the challenge of Π in the first round of Π^{WI}. Subsequently, $\mathcal{P}$ and $\mathcal{V}$ play Σ-protocol Π where at the second round (corresponding to the 3 round of Π^{WI}), $\mathcal{V}$ decommits to the challenge.

To prove soundness we need to construct an adversary $\mathcal{A}$ that plays against $\mathcal{P}^\star$ and opens the commitment in more than one way so that we can use the special soundness of Π to reach a contradiction. Instead, to prove rWI we need the commitment to be binding for $\mathcal{V}^\star$ so that reset attacks are ineffective.

We would thus like to use the notion of a simulatable commitment of [MP03] in which the commitment is not opened but the sender announces the decommitted value and then sender and receiver engage in a Σ-protocol Π_{MP} in which the sender proves that the announced value is the value committed. In order to let $\mathcal{A}$ open the commitment in two ways while still preserving binding against $\mathcal{V}^\star$, we could use our OR-composition technique combining Π_{MP} with Schnorr's protocol, obtaining Π^{OR} that lets the sender prove that either the announced value is correct or it knows the DLog of a challenge sent by the receiver. $\mathcal{A}$ can break the DLog challenge running in super-polynomial time therefore succeeding (before and after a rewind) in announcing two different values m, m' that are different from the committed one. Instead $\mathcal{V}^\star$ will not be able to decommit to a different message since otherwise we could use $\mathcal{V}^\star$ to break in polynomial time the DLog challenge.

There is a caveat though. Our OR-composition technique requires at least one statement to be known to the prover of Π^{OR} (i.e., $\mathcal{V} \in \Pi^{\mathsf{WI}}$) when the protocol starts. Clearly the DLog challenge will arrive only at the second round, therefore the known statement must be that the commitment corresponds to a message m. However the first rounds of Π^{OR} will be based on such a statement that therefore can not be changed later and this makes $\mathcal{A}$ stuck on m.

The simultaneous needs of allowing $\mathcal{A}$ to switch from m to m' and of using Π^{OR} on an fixed statement forces us to use one more tool. We are again inspired by the simulatable commitment of [MP03] since it is shown in [MP03] that the special HVZK simulator also works with false statements (see the notion of simulation of "random bad commitments" in [MP03]) by deviating with the distribution of the first round of Π_{MP} and, of course, still leaving the transcript indistinguishable. We will make use of such trick by letting $\mathcal{A}$ deviate in the same way, and applying our OR-composition technique to obtain Π^{OR} using as statements the distribution of the first round of Π_{MP} and the DLog challenge.

Putting all pieces together, we have that: 1) $\mathcal{A}$ is able to announce m first and m' later by deviating when computing the first round of Π_{MP}; 2) $\mathcal{A}$ is able to succeed in Π^{OR} despite having deviated in Π_{MP} (this holds because $\mathcal{A}$ will break the DLog challenge); 3) Π^{OR} is an OR composition of two statements such that one is fixed (i.e., the first round of Π_{MP}) when Π^{OR} starts. 4) $\mathcal{V}^\star$ can announce only the actually committed message since Π^{OR} forces him to run Π_{MP} correctly (we would break the DLog challenge in polynomial time otherwise).

Let us now proceed more formally. Let Π^{dlog} be an input-delayed Σ-protocol for the discrete log polynomial-time relation

$$\mathsf{DLOG} = \{((\mathcal{G}, q, g, Y), y) : g^y = Y\}$$

over groups $\mathcal{G}$ of prime-order q; e.g., Schnorr's Σ-protocol [Sch89].

Consider also the following polynomial-time relation

$$\mathsf{DDH} = \Big\{\big(((\mathcal{G}, q, g), A = g^a, B = g^b, C = g^{ab}), b\big) : A^b = C\Big\}.$$

over groups $\mathcal{G}$ of prime-order q. In the rest of the paper we will refer to a tuple $((\mathcal{G}, q, g), A = g^a, B = g^b, C = g^{ab})$ as *DH-tuple*.

We now consider the protocol Π^{OR}, with the associated algorithm $(\mathsf{P}_1^{\mathsf{OR}}, \mathsf{P}_2^{\mathsf{OR}}, \mathsf{V}^{\mathsf{OR}})$, for the following polynomial-time relation

$$\mathcal{R}_{\mathsf{OR}} = \Big\{((x_0, x_1), w) : \big((x_0, w) \in \mathsf{DLOG} \wedge x_1 \in \hat{L}_{\mathsf{DDH}}\big)\mathsf{OR}$$
$$\big((x_1, w) \in \mathsf{DDH} \wedge x_0 \in \hat{L}_{\mathsf{DLOG}}\big)\Big\}.$$

We now denote by l^{OR} the challenge length of the Σ-protocol Π^{OR}.

We denote by x the common input and by w the witness received by $\mathcal{P}$ s.t. $(x, w) \in \mathcal{R}$. We also consider a Σ-protocol Π, with the associated algorithm $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$ for the polynomial-time relation $\mathcal{R}$.

Security parameters λ and $\hat{\lambda}$ will be determined as part of the proof.

The security of our proposed rWI $\Pi^{\mathsf{WI}} = (\mathcal{P}, \mathcal{V})$ for $\mathcal{R}$ is based on the well-known DDH assumption and on a strengthening of it (see Assumption 4).

Protocol 6. *(4-round efficient resettable WI from Σ-protocols)*
Public input: instance x and security parameter 1^λ.
Private input to $\mathcal{P}$: witness w such $(x, w) \in \mathcal{R}$.
Round 1: from $\mathcal{V}$ to $\mathcal{P}$.

1.1 Committing to the challenge c of Π

$\mathcal{V}$ randomly selects $(\mathcal{G}, q, g) \leftarrow \mathsf{IG}(1^\lambda)$ and $(\hat{\mathcal{G}}, \hat{q}, \hat{g}) \leftarrow \mathsf{IG}(1^{\hat{\lambda}})$ (the actual value of $\hat{\lambda}$ is determined in the proof of soundness) and sends them to $\mathcal{P}$;

$\mathcal{V}$ randomly selects $c \leftarrow \mathbb{Z}_q$ and commits to it by randomly selecting $r_1 \leftarrow \mathbb{Z}_q$ and $h \leftarrow \mathcal{G}$ and setting $g_1 = g^{r_1}$ and $h_1 = h^{r_1+c}$;

$\mathcal{V}$ sends (g_1, h_1) to $\mathcal{P}$;

1.2 Preparing first input for Π^{OR}.

$\mathcal{V}$ randomly selects $r_2 \leftarrow \mathbb{Z}_q$ and computes (g_2, h_2) by setting $g_2 = g^{r_2}$ and $h_2 = h^{r_2}$;

$\mathcal{V}$ sends $x_0^{\mathsf{OR}} = (\mathcal{G}, g, h, g_2, h_2)$ to $\mathcal{P}$;

1.3 Computing first round of Π^{OR}.

$\mathcal{V}$ randomly selects coin tosses R_1, sets $a^{\mathsf{OR}} = \mathsf{P}_1^{\mathsf{OR}}(x_0^{\mathsf{OR}}; R_1)$ and sends a^{OR} to $\mathcal{P}$;

Round 2: from $\mathcal{P}$ to $\mathcal{V}$.

2.1 Picking randomness.

$\mathcal{P}$ reads ω from its random tape, computes $\bar{R} = F_\omega(x, x_0^{\mathsf{OR}}, (g_1, h_1), (\hat{\mathcal{G}}, \hat{q}, \hat{g}), a^{\mathsf{OR}})$ and parses it as $\bar{R} = (R_2, R_3)$;

2.2 Computing first round of Π.

$\mathcal{P}$ sets $a = \mathsf{P}_1(x, w; R_2)$ and sends it to $\mathcal{V}$;

2.3 Preparing second input for Π^{OR} and sending the challenge for Π^{OR}.

$\mathcal{P}$ randomly selects $y \leftarrow \mathbb{Z}_{\hat{q}}$, sets $Y = \hat{g}^y$ and sends it to $\mathcal{V}$;

$\mathcal{P}$ randomly selects $c^{\mathsf{OR}} \leftarrow \{0,1\}^{l^{\mathsf{OR}}}$ and sends it to $\mathcal{V}$;

Round 3: from $\mathcal{V}$ to $\mathcal{P}$.

3.1 Opening commitment of challenge for Π.

$\mathcal{V}$ sends c to $\mathcal{P}$;

$\mathcal{V}$ randomly selects challenge $c_1 \leftarrow \mathbb{Z}_q$ and sets $z_1 = r_2 + c_1 \cdot r_1$;

$\mathcal{V}$ sends c_1, z_1 to $\mathcal{P}$;

3.2 Computing third round of Π^{OR}.

$\mathcal{V}$ sets $x_1^{\mathsf{OR}} = (\hat{q}, \hat{g}, Y)$, $w^{\mathsf{OR}} = r_2$ and computes and sends $z^{\mathsf{OR}} \leftarrow \mathsf{P}_2^{\mathsf{OR}}((x_0^{\mathsf{OR}}, x_1^{\mathsf{OR}}), c^{\mathsf{OR}}, w^{\mathsf{OR}}, R_1)$;

Round 4: from $\mathcal{P}$ to $\mathcal{V}$.

4.1 Verification of Π^{OR}.

If $\mathsf{V}^{\mathsf{OR}}((x_0^{\mathsf{OR}}, x_1^{\mathsf{OR}}), a^{\mathsf{OR}}, c^{\mathsf{OR}}, z^{\mathsf{OR}}) = 0$, $\mathcal{P}$ aborts;

4.2 Checking the decommitment.

If $g^{z_1} \neq g_2 \cdot g_1^{c_1}$ or $h^{z_1} \neq h_2 \cdot h_1^{c_1} \cdot h^{-c \cdot c_1}$, $\mathcal{P}$ aborts;

4.3 Compute third round of Π.

$\mathcal{P}$ computes $z \leftarrow \mathsf{P}_2(x, w, c, R_2; R_3)$ and sends it to $\mathcal{V}$;

$\mathcal{V}$'s decision: $\mathcal{V}$ accepts if and only if $\mathsf{V}(x, a, c, z) = 1$.

We observe that values (g_2, h_2) computed at the Step 1.2 by the $\mathcal{V}$ are actually the first round of the Σ-protocol Π_{MP}, as described in the informal section. The transcript for that protocol is completed by $\mathcal{V}$ at the steps 3.1 by computing the values c_1 and z_1. In the end, at the step 4.2, the prover $\mathcal{P}$ verifier that the transcript is accepting for the protocol Π_{MP} with respect to a committed message c.

Completeness is straightforward.

Soundness. Let $\mathcal{P}^\star$ be an efficient algorithm and suppose that, for $x \notin L_{\mathcal{R}}$, $\mathcal{P}^\star$ makes the honest verifier $\mathcal{V}$ accept. We consider the following three hybrid experiments $\mathcal{H}_0, \mathcal{H}_1$, and $\mathcal{H}_2$.

The first hybrid experiment $\mathcal{H}_0(x)$ is simply the real game between of $\mathcal{P}^\star$ and the honest verifier $\mathcal{V}$.

In the second hybrid experiment, $\mathcal{H}_1(x)$, $\mathcal{P}^\star$ interacts with a verifier $\mathcal{V}_1$ that, upon receiving Y, computes its discrete log y and sets $w^{\mathsf{OR}} = y$ at Step 3.2. Therefore, $\mathcal{V}_1$ uses y instead of r_2 to compute message z^{OR}. $\mathcal{H}_0(x)$ and $\mathcal{H}_1(x)$ are indistinguishable because of the perfect adaptive WI of Π^{OR}.

In hybrid $\mathcal{H}_2(x)$, $\mathcal{P}^\star$ interacts with $\mathcal{V}_2$ that differs from $\mathcal{V}_1$ because it selects c and c' at Step 1.1, commits to c' and sends c at Step 3.1. More formally, we describe below Round 1 and 3 of $\mathcal{V}_2$.

Round 1: from $\mathcal{V}_2$ to $\mathcal{P}^\star$.

1.1 Committing to the challenge c' of Π.

$\mathcal{V}_2$ randomly selects $(\mathcal{G}, q, g) \leftarrow \mathsf{IG}(1^\lambda)$, $(\hat{\mathcal{G}}, \hat{q}, \hat{g}) \leftarrow \mathsf{IG}(1^{\hat{\lambda}})$, $c, \underline{c'}, r_1 \leftarrow \mathbb{Z}_q$, and $h \leftarrow \mathcal{G}$ and sets $g_1 = g^{r_1}$ and $\underline{h_1 = h^{r_1 + c'}}$;

$\mathcal{V}_2$ sends $(\mathcal{G}, q, g, h)$, $(\hat{\mathcal{G}}, \hat{q}, \hat{g})$ and (g_1, h_1) to $\mathcal{P}^\star$;

1.2 Preparing first input for Π^{OR}.

$\mathcal{V}_2$ randomly selects $z_1, c_1 \leftarrow \mathbb{Z}_q$ and sets $\underline{g_2 = g^{z_1} \cdot g_1^{-c_1}}$ and $\underline{h_2 = h^{z_1 + c \cdot c_1} \cdot h_1^{-c_1}}$;

$\mathcal{V}_2$ sends $x_0^{\mathsf{OR}} = (\mathcal{G}, g, h, g_2, h_2)$ to $\mathcal{P}^\star$;

1.3 Compute first round of Π^{OR}.

$\mathcal{V}_2$ randomly selects coin tosses R_1, sets $a^{\mathsf{OR}} = \mathsf{P}_1^{\mathsf{OR}}(x_0^{\mathsf{OR}}; R_1)$ and sends a^{OR} to $\mathcal{P}^\star$;

Round 3: from $\mathcal{V}_2$ to $\mathcal{P}^\star$.

3.1 Open commitment of challenge for Π

$\mathcal{V}_2$ sends c to $\mathcal{P}^\star$;

$\mathcal{V}_2$ sends c_1, z_1 to $\mathcal{P}^\star$;

3.2 Compute third round of Π^{OR}.

$\mathcal{V}_2$ sets $x_1^{\mathsf{OR}} = (\hat{q}, \hat{g}, Y)$, $w^{\mathsf{OR}} = y$ such that $\hat{g}^y = Y$ and computes and sends $z^{\mathsf{OR}} \leftarrow \mathsf{P}_2^{\mathsf{OR}}((x_0^{\mathsf{OR}}, x_1^{\mathsf{OR}}), c^{\mathsf{OR}}, w^{\mathsf{OR}}, R_1)$ to $\mathcal{P}^\star$;

We next show that, under the DDH assumption, the two hybrids are indistinguishable and we will do so by describing a simulator $\mathcal{B}$ that, on input a tuple $T = ((\mathcal{G}, q, g), h, A, B)$, outputs the view of $\mathcal{P}^\star$ in $\mathcal{H}_1$ or $\mathcal{H}_2$, depending on whether T is DH or not. Specifically, $\mathcal{B}$ executes the code of $\mathcal{V}_2$ with the following modification: at step 1.1, g_1 is set equal to A and h_1 is set equal to $B \cdot h^c$.

Suppose now that T is a DH tuple; that is, there exists $r_1 \in \mathbb{Z}_q$ such that $A = g^{r_1}$ and $B = h^{r_1}$. Then it is easy to see that g_1 and h_1 are distributed exactly like in $\mathcal{H}_1$. Moreover, g_2 and h_2 have the same discrete log with respect to g and h and c_1 is random in $\mathbb{Z}_p$ and z_1 passes the verifications in Step 4.2 of the honest prover. We can thus conclude that if T is DH then $\mathcal{B}$'s output has the same distribution as the view of $\mathcal{V}^\star$ in $\mathcal{H}_1$. Finally, suppose that T is not a DH tuple; that is there exist $r_1 \neq r_1' \in \mathbb{Z}_q$ such that $A = g^{r_1}$ and $B = h^{r_1'}$. Then g_1 and h_1 are distributed as in $\mathcal{H}_2$ with $c' = r_1' - r_1 + c$.

The running time of the simulator $\mathcal{B}$ is dominated by the time needed to compute the discrete log of Y which is $O(2^{\hat{\lambda}})$. Therefore we choose $\hat{\lambda}$ small enough so that deciding whether a tuple with security parameter λ is DH in time $O(2^{\hat{\lambda}})$ constitutes a contradiction of the DDH assumption.

We are now ready to show that it is possible to extract (in sub-exponential time) a witness w for $x \notin L_{\mathcal{R}}$ from $\mathcal{P}^\star$ with a positive probability thus reaching a contradiction. The extractor E impersonates the verifier and interacts with $\mathcal{P}^\star$ in the following way.

Round 1: from E to $\mathcal{P}^\star$.

1.1 Committing to the challenge c' of Π.

E randomly selects $(\mathcal{G}, q, g) \leftarrow \mathsf{IG}(1^\lambda)$, $(\hat{\mathcal{G}}, \hat{q}, \hat{g}) \leftarrow \mathsf{IG}(1^{\hat{\lambda}})$, $c', r_1 \leftarrow \mathbb{Z}_q$, and $h \leftarrow \mathcal{G}$ and sets $g_1 = g^{r_1}$ and $h_1 = h^{r_1 + c'}$;

E sends $(\mathcal{G}, q, g, h)$, $(\hat{\mathcal{G}}, \hat{q}, \hat{g})$ and (g_1, h_1) to $\mathcal{P}^\star$;

1.2 Preparing first input for Π^{OR}.

E randomly selects $r_2, r_2' \leftarrow \mathbb{Z}_q$, sets $g_2 = g^{r_2}$ and $h_2 = h^{r_2'}$;

E sends $x_0^{\mathsf{OR}} = (\mathcal{G}, g, h, g_2, h_2)$ to $\mathcal{P}^\star$;

1.3 Compute first round of Π^{OR}.

E randomly selects coin tosses R_1, sets $a^{\mathsf{OR}} = \mathsf{P}_1^{\mathsf{OR}}(x_0^{\mathsf{OR}}; R_1)$ and sends a^{OR} to $\mathcal{P}^\star$;

Round 3: from E to $\mathcal{P}^\star$.

3.1 Open commitment of challenge for Π.

E randomly selects $c \leftarrow \mathbb{Z}_q$ and sends it to $\mathcal{P}^\star$;

E sets $c_1 = (c - c')/(r'_2 - r_2)$ and $z_1 = r_2 + r_1 \cdot c_1$;

E sends c_1, z_1 to $\mathcal{P}^\star$;

3.2 Compute third round of Π^{OR}.

E sets $x_1^{\mathsf{OR}} = (\hat{q}, \hat{g}, Y)$, $w^{\mathsf{OR}} = y$ and computes and sends $z^{\mathsf{OR}} \leftarrow \mathsf{P}_2^{\mathsf{OR}}((x_0^{\mathsf{OR}}, x_1^{\mathsf{OR}}), c^{\mathsf{OR}}, w^{\mathsf{OR}}, R_1)$;

3.3 receive z from $\mathcal{P}^\star$;

3.4 Rewind $\mathcal{P}^\star$.

Rewind $\mathcal{P}^\star$ exactly to the state before Round 3 is started;

3.1R Re-open commitment of challenge for Π

E randomly selects $\hat{c} \leftarrow \mathbb{Z}_q$ and sends it to $\mathcal{P}^\star$;

E sets $\hat{c}_1 = (\hat{c} - c')/(r'_2 - r_2)$ and $\hat{z}_1 = r_2 + r_1 \cdot c_1$;

E sends $\hat{c}_1, \hat{z}_1$ to $\mathcal{P}^\star$;

3.2R Compute third round of Π^{OR}.

E sets $x_1^{\mathsf{OR}} = (\hat{q}, \hat{g}, Y)$, $w^{\mathsf{OR}} = r_2$ and computes and sends $z^{\mathsf{OR}} \leftarrow \mathsf{P}_2^{\mathsf{OR}}((x_0^{\mathsf{OR}}, x_1^{\mathsf{OR}}), c^{\mathsf{OR}}, w^{\mathsf{OR}}, R_1)$;

3.3R receive $\hat{z}$ from $\mathcal{P}^\star$;

3.5 use z and $\hat{z}$ to extract a witness w for x;

We make the following two observations. First of all, in both interactions of E with $\mathcal{P}^\star$ (the one before the rewind and the one after the rewind), $\mathcal{P}^\star$ sees exactly the same distribution as in $\mathcal{H}_2$. Therefore for each of the two interactions, $\mathcal{P}^\star$ has a non-negligible probability of making the verifier accept. This implies that with non-negligible probability (a, c, z) and $(a, \hat{c}, \hat{z})$ are a collision for x and therefore, by the properties of the Σ-protocol Π, it is possible to extract (in super-polynomial time) a witness for $x \in L_{\mathcal{R}}$. Contradiction.

Proof of rWI. The idea of the proof for rWI is very simple. We consider the prover $\mathcal{P}_1$ that, when instructed to use randomness with index j and input with index i and receives first message $\mathtt{msg}$, checks first if a tuple $(j, i, \mathtt{msg}, R)$ has been stored in a previous step. If such a tuple is found then R is used as source of randomness; otherwise, a fresh R is selected and tuple $(j, i, \mathtt{msg}, R)$ is stored. Clearly, by pseudo-randomness, $\mathcal{P}_1$ is indistinguishable from the honest prover $\mathcal{P}$.

Now we observe that the resetting verifier $\mathcal{V}^\star$ is performing an attack on $\mathcal{P}_1$ in which two distinct interactions use the same randomness iff they share j, the input and the first message. More precisely, we say that interactions t_1 and t_2 between $\mathcal{P}_1$ and $\mathcal{V}^\star$ are a *collision* if they share the input, the randomness used by $\mathcal{P}_1$ and the first message and $\mathcal{V}^\star$ opens the commitment in the first messages in two different ways. If $\mathcal{V}^\star$ has a non-negligible probability of producing a transcript then we can break the Strong DLog Assumption (see Definition 2). Consider algorithm DL that receives as input $(\bar{x}, \bar{w}^0, \bar{w}^1)$ for which $\mathcal{V}^\star$ distinguishes $\mathcal{H}_2(\bar{x}, \bar{w}^0)$ from $\mathcal{H}_2(\bar{x}, \bar{w}^1)$. DL interacts with $\mathcal{V}^\star$ and at the start of the interaction guesses two interactions $t_1 < t_2$ (in the hope they constitute a collision). In all interactions other than t_1 and t_2, DL runs just like $\mathcal{P}_1$. When DL receives the discrete log parameters from $\mathcal{V}^\star$ as part of the first message of interaction t_1, it forwards them to the challenger of the discrete log and receives Y (and the task is to compute the discrete log of Y) and uses it as part of the second message of interaction t_1. When interaction t_2 is activated DL checks if the first message is the same as the first message of interaction t_1. If this is the case (and this happens with non-negligible probability) DL continues and sends the same second message, including Y. Notice that $\mathcal{V}^\star$ expects to receive the same message since it thinks it is interacting with $\mathcal{P}_1$ that is using the same randomness. Otherwise, DL aborts. Then DL rewinds $\mathcal{V}^\star$ and sends a different challenge in each of the two interactions. In at least one of them $\mathcal{V}^\star$ has opened the commitment to a different message than the one committed to by the commitment in the first message. This means that $\mathcal{V}^\star$ has used knowledge of the discrete log of Y to complete the interaction and thus DL can extract it.

Finally, let us consider the case in which $\mathcal{V}^\star$ produces a collision in his resetting attack only with negligible probability. This means that $\mathcal{V}^\star$ is conducting a successful concurrent WI attack on the argument system. Standard arguments show that this contradicts the WI of Π[9].

6.3 Efficient 4-Round Resettable Zero Knowledge in the BPK model

Let $\mathcal{R}$ be a polynomial-time relation with Σ-protocol $\Pi = (\mathcal{P}, \mathcal{V})$. In this section we give an informal description of an efficient 4-round argument system (Π^{BPK}) for $\mathcal{R}$, that is resettable zero knowledge and concurrently sound in the BPK model.

In our protocol we consider that each entry of the verifier's identities file is an element of a group $\mathcal{G}$ in which is hard to compute the discrete logarithm. $\Pi^{\mathsf{BPK}} = (\mathcal{P}, \mathcal{V})$ for $\mathcal{R}$ consists of the interleaved execution of two protocol: Π^{WI}, in which $\mathcal{P}$ acts a prover, and Π^{OR}, in which $\mathcal{V}$ acts as a prover. Π^{OR} is the Σ-protocol obtained from the OR composition of two Schnorr's Σ-protocols Π_0 and Π_1. Π^{OR} is used by the verifier $\mathcal{V}$ to prove that she knows either the discrete logarithm of a selected identity $\mathtt{id}$, or the discrete logarithm of an elements sent by the prover $\mathcal{P}$ at the second round of Π^{BPK}. Π^{WI} is used by the prover $\mathcal{P}$ to show that she know either the witness for the theorem x to be proved or the discrete logarithm of $\mathtt{id}$.

To prove concurrent soundness of Π^{BPK} we use the same arguments of Section 6.2, with the difference that at the end of the proof we can extract (in sub-exponential time) the discrete logarithm of $\mathtt{id}$ (instead of a witness for x), and construct a reduction using complexity leveraging to break the assumption that it is hard to compute the discrete logarithm of an elements in $\mathcal{G}$.

To prove $\mathsf{r}\mathcal{ZK}$ we consider a simulator $\mathcal{B}$ that rewinds the verifier $\mathcal{V}$ to get the discrete logarithm of $\mathtt{id}$ (used as a witness by $\mathcal{V}$ to run Π^{OR}) and uses this witness to complete the execution of the protocol Π^{WI}. We observe that $\mathcal{B}$ works correctly only if the first round of Π^{WI} can be computed without using the witness. In this case we have no problem, because we can construct Π^{WI} starting form a Σ-protocol Π' that enjoys this property. More specifically, Π' is the result of an OR compo-

[9] We are implicitly using the fact that the Σ-protocol Π is WI. This is certainly true if Π is perfect [CDS94]. If Π is only computational then we consider self OR composition of Π which by [GMY06] is WI.

sition (using [CDS94]) of Π and of a Schnorr's Σ-protocol that is delayed witness. It is easy to see that the property of delayed witness of Schnorr's Σ-protocol holds even in Π' and in turn in Π^{WI}. The last observation make us able to conclude the proof sketch.

7 Acknowledgments

We thank Berry Schoenmakers for various useful discussions on Σ-protocols.

The work of the third author was supported by the MACS project under NSF Frontier grant CNS-1414119 and by NSF grant 1012798. This work was done in part while the third author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant CNS-1523467.

Part of this work will appear in the proceedings of the Theory of Cryptography Conference (TCC) 2016-A [CPS+16].

References

[AOS13] Masayuki Abe, Tatsuaki Okamoto, and Koutarou Suzuki. Message recovery signature schemes from sigma-protocols. *IEICE Transactions*, 96-A(1):92–100, 2013.

[BFGM01] Mihir Bellare, Marc Fischlin, Shafi Goldwasser, and Silvio Micali. Identification protocols secure against reset attacks. In *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 495–511. Springer, 2001.

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, pages 390–420, 1992.

[Blu86] Manuel Blum. How to prove a theorem so no one else can claim it. In *International Congress of Mathematicians*, page 1444, 1986.

[BP15] Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 401–427, 2015.

[BPSV08] Carlo Blundo, Giuseppe Persiano, Ahmad-Reza Sadeghi, and Ivan Visconti. Improved security notions and protocols for non-transferable identification. In *Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings*, volume 5283 of *Lecture Notes in Computer Science*, pages 364–378, 2008.

[CD98] Ronald Cramer and Ivan Damgård. Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free? In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 424–441. Springer, 1998.

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Extractable perfectly one-way functions. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, pages 449–460, 2008.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In YvoG. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer Berlin Heidelberg, 1994.

[CDV06] Dario Catalano, Yevgeniy Dodis, and Ivan Visconti. Mercurial commitments: Minimal assumptions and efficient constructions. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 120–144. Springer, 2006.

[CG15] Pyrros Chaidos and Jens Groth. Making sigma-protocols non-interactive without random oracles. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pages 650–670, 2015.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 235–244, 2000.

[CPS$^+$16] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Improved OR composition of sigma-protocols. In *Theory of Cryptography - 13th Theory of Cryptography Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings*, 2016.

[CPSV15] Michele Ciampi, Giuseppe Persiano, Luisa Siniscalchi, and Ivan Visconti. A transform for nizk almost as efficient and general as the fiat-shamir transform without programmable random oracles. *IACR Cryptology ePrint Archive*, 770, 2015.

[CPSV16] Michele Ciampi, Giuseppe Persiano, Luisa Siniscalchi, and Ivan Visconti. A transform for NIZK almost as efficient and general as the fiat-shamir transform without programmable random oracles. In *Theory of Cryptography - 13th Theory of Cryptography Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, Proceedings*, 2016.

[Cra96] Ronald Cramer. *Modular design of secure yet practical cryptographic protocols*. PhD thesis, University of Amsterdam, 1996.

[CV05a] Dario Catalano and Ivan Visconti. Hybrid trapdoor commitments and their applications. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, pages 298–310, 2005.

[CV05b] Giovanni Di Crescenzo and Ivan Visconti. Concurrent zero knowledge in the public-key model. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 816–827. Springer, 2005.

[CV07] Dario Catalano and Ivan Visconti. Hybrid commitments and their applications to zero-knowledge proof systems. *Theor. Comput. Sci.*, 374(1-3):229–260, 2007.

[Dam10] Ivan Damgård. On Σ-protocol. http://www.cs.au.dk/~ivan/Sigma.pdf, 2010.

[DG03] Ivan Damgård and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 426–437, 2003.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 283–293, 2000.

[DPV04] Giovanni Di Crescenzo, Giuseppe Persiano, and Ivan Visconti. Constant-round resettable zero knowledge with concurrent soundness in the bare public-key model. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 237–253. Springer, 2004.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 308–317. IEEE Computer Society, 1990.

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 253–280, 2015.

[GMY06] Juan A. Garay, Philip MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. *Journal of Cryptology*, 19(2):169–209, 2006.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 339–358, 2006.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In Christoph G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128. Springer, 1988.

[GRRV14] Vipul Goyal, Silas Richelson, Alon Rosen, and Margarita Vald. An algebraic approach to non-malleability. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 41–50. IEEE Computer Society, 2014.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology-Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 335–354, 2004.

[Lin15] Yehuda Lindell. An efficient transform from Sigma protocols to NIZK with a CRS and non-programmable random oracle. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, pages 93–109, 2015.

[LP15] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. *J. Cryptology*, 28(2):312–350, 2015.

[LS90] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *Advances in Cryptology - CRYPTO*, 1990.

[Mau09] Ueli M. Maurer. Unifying zero-knowledge proofs of knowledge. In *Progress in Cryptology - AFRICACRYPT 2009, Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings*, pages 272–286, 2009.

[Mau15] Ueli Maurer. Zero-knowledge proofs of knowledge for group homomorphisms. *Designs, Codes and Cryptography*, pages 1–14, 2015.

[MP03] Daniele Micciancio and Erez Petrank. Simulatable commitments and efficient concurrent zero-knowledge. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, pages 140–159, 2003.

[OPV10] Rafail Ostrovsky, Omkant Pandey, and Ivan Visconti. Efficiency preserving transformations for concurrent non-malleable zero knowledge. In *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, pages 535–552, 2010.

[ORV14] Rafail Ostrovsky, Vanishree Rao, and Ivan Visconti. On selective-opening attacks against encryption schemes. In *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, volume 8642 of *Lecture Notes in Computer Science*, pages 578–597. Springer, 2014.

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 160–176. Springer, 2003.

[Pas04] Rafael Pass. Alternative variants of zero-knowledge proofs. Master's thesis, Kungliga Tekniska Högskolan, 2004. Licentiate Thesis Stockholm, Sweden.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 129–140, 1991.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, pages 387–398, 1996.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer New York, 1989.

[SV12] Alessandra Scafuro and Ivan Visconti. On round-optimal zero knowledge in the bare public-key model. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 153–171, 2012.

[Vis06] Ivan Visconti. Efficient zero knowledge on the internet. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pages 22–33, 2006.

[YZ07] Moti Yung and Yunlei Zhao. Generic and practical resettable zero-knowledge in the bare public-key model. In *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, pages 129–147, 2007.

A More About Σ-Protocols

Theorem 13. *For every relation $\mathcal{R}$ such that $L_{\mathcal{R}} \notin \mathsf{BPP}$ there exist Σ-protocols that are not WI.*

Proof. Let $\Pi' = (\mathcal{P}', \mathcal{V}')$ be a Σ-protocol for the relation $\mathcal{R}$ with challenge length l and let $(\mathsf{P}'_1, \mathsf{P}'_2, \mathsf{V}')$ be the triple of PPT algorithms associated to Π'. We use these algorithms to describe a Σ-protocol Π with associated algorithms $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$ that is not WI. Consider $(x, w) \in \mathcal{R}$.

1. P_1 on input (x, w) and randomness R_1 parses it as (r_1, c_p) where c_p is an l-bit string, computes $a' \leftarrow \mathsf{P}'_1(x, w; r_1)$, and outputs $a = (a', c_p)$.
2. P_2, on input (x, w), R_1, a challenge c computes $z' \leftarrow \mathsf{P}'_2(x, w, r_1, c)$ and if $c = c_p$ then it also sets $z = w$ otherwise it sets $z = z'$; finally it outputs z.
3. V, on input x, $a = a', c_p$, c and z, makes the following steps: in case c is different from c_p it outputs $\mathsf{V}'(x, a', c, z)$ otherwise it output 1 iff $(x, z) \in \mathcal{R}$.

We now check that Π is a Σ-protocol.

- **Completeness**: The completeness of Π follows from the completeness of Π' except when c is equal to c_p. In this case $\mathcal{P}$ has a witness and sends it to $\mathcal{V}$ that still accepts.
- **Special Soundness**: $\mathsf{Extract}$ on input a collision $(a = (a', c_p), c_1, z_1)$ $(a = (a', c_p), c_2, z_2)$ works as follows:
  - if c_1 and c_2 are different from c_p then it runs the extractor $\mathsf{Extract}'$ of Π' on input x and a collision (a', c_1, z_1) (a', c_2, z_2) returning its output.
  - if c_1 is equal to c_p, it outputs z_1 while instead if c_2 is equal to c_p it outputs z_2.
- **SHVZK** $\mathsf{Sim}(x, c)$ of Π works as follows:
  - computes $(a', z') \leftarrow \mathsf{Sim}'(x, c)$, where $\mathsf{Sim}'(x, c)$ is the simulator of Π';
  - picks $c_p \leftarrow \{0, 1\}^l$.
  - if c_p is equal to c then it aborts, otherwise it outputs $(a = (a', c_p), z')$.

We prove that Π is computational SHVZK, namely: for any l-bit string c, the transcript given in output by $\mathsf{Sim}(x, c)$ is computationally indistinguishable from a honest transcript where the challenge is c and $\mathcal{P}$ runs on common input x and private input w such that $(x, w) \in \mathcal{R}$.

Suppose there exists a distinguisher $\mathcal{A}$ for the SHVZK of Π, then we can show a distinguisher $\mathcal{A}'$ for the SHVZK of Π'.

$\mathcal{A}'$ runs $\mathcal{A}$ that outputs a pair (x, w) and a challenge c. $\mathcal{A}'$ then asks the challenger of SHVZK to produce a transcript (either honest or simulated) for instance x, witness w and challenge c. $\mathcal{A}'$ obtains from the challenger a pair (a', z') such that (a', c, z') is an accepting transcript. $\mathcal{A}'$ picks randomly an l-bit string c', sets $a = a'|c'$, $z = z'$, feeds (a, z) to $\mathcal{A}$, and outputs what $\mathcal{A}$ outputs.

We note that the success probability of $\mathcal{A}'$ is statistically close to the one of $\mathcal{A}$ since the probability that c is equal to c' is negligible and this case is the only deviation among the two distributions.

We finally note that Π' is not WI since an adversarial verifier $\mathcal{V}^\star$ can obtain a witness by just sending a challenge c that is equal to c_p. As a consequence $\mathcal{V}^\star$ can get and output a witness for $x \in L$ during an execution with $\mathcal{P}$. Clearly no PPT simulator can produce the same output unless $L \in \mathsf{BPP}$. $\square$

A.1 Challenge Length of Σ-Protocols

In this section we show how one can reduce or stretch the size of the challenge in a Σ-protocol and in a $\tilde{\Sigma}$-protocol.

Challenge-length amplification. The challenge of a Σ-protocol can be extended through parallel repetition.

Lemma 1. *[CDS94, Dam10] Let Π be a Σ-protocol (resp. $\tilde{\Sigma}$-protocol) for relation $\mathcal{R}$ and challenge length l. Running Π k-times in parallel for the same instance x corresponds to running Σ-protocol (resp. $\tilde{\Sigma}$-protocol) for $\mathcal{R}$ with challenge length $k \cdot l$.*

Challenge-length reduction.

Lemma 2. *[Dam10] Given a Σ-protocol of challenge length l for the relation $\mathcal{R}$, is possible to construct a Σ-protocol, for the same relation $\mathcal{R}$ with challenge length l' where $l' < l$.*

We now show that Lemma 2 it is true even when we consider a $\tilde{\Sigma}$-protocol. One possibility to obtain this result is to convert the $\tilde{\Sigma}$-protocol in a Σ-protocol, and then use Lemma 2. We show how to obtain the same result without first converting the $\tilde{\Sigma}$-protocol to a Σ-protocol.

Lemma 3. *For any $\tilde{\Sigma}$-protocol $\Pi = (\mathcal{P}, \mathcal{V})$, for a relation $\mathcal{R}$ with challenge length l, simulator Sim, and the associated triple $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$, there exists a $\tilde{\Sigma}$-protocol $\Pi' = (\mathcal{P}', \mathcal{V}')$, for the same relation $\mathcal{R}$, with challenge length l', where $l' < l$ and with the same efficiency.*

Proof. We show Π' by presenting the associated triple $(\mathsf{P}'_1, \mathsf{P}'_2, \mathsf{V}')$ of efficient PPT algorithms.

1. P'_1 on input (x, w) and randomness R_1 computes and outputs $a \leftarrow \mathsf{P}_1(x, w; R_1)$.
2. P'_2 on input (x, w), $c \in \{0, 1\}^{l'}$, R_1 and randomness R_2, parses R_2 as (pad, R'_2) where pad is an $(l - l')$-bit string, sets $c' = c|pad$, computes $z \leftarrow \mathsf{P}_2(x, w, R_1, c'; R'_2)$ and outputs $z' = (z, pad)$.
3. V' on input $x, a, z' = (z, pad)$ and c, outputs the output of $\mathsf{V}(a, c|pad, z)$.

Completeness follows directly from the completeness of Π.

HVZK We can consider the simulator Sim', that on input x runs as follows:

- runs $(a, c, z) \leftarrow \mathsf{Sim}(x)$;
- sets pad equal to the last $l - l'$ bits of c, and sets c' equal to the fist l' bits of c;
- outputs $(a, c', (z, pad))$.

Special soundness follows directly from the special soundness of Π. □

From Lemma 1, 2 and 3, we can claim the following theorem.

Theorem 14. *Suppose that relation $\mathcal{R}$ has a Σ-protocol ($\tilde{\Sigma}$-protocol) Π. Then, for any challenge length l, $\mathcal{R}$ admits a Σ-protocol ($\tilde{\Sigma}$-protocol) Π' with challenge length l'. If $l' \leq l$ than Π' is almost as efficient as Π. Otherwise the communication and computation complexities of Π' are l'/l times the ones of Π.*

B Pre-Image Protocol

In this section we described the pre-image protocol for proving knowledge of a pre-image of a value in the range of a homomorphic function. This protocol is an abstraction of a large class of protocols like Schnorr's [Sch89] protocol and Guillou-Quisquater [GQ88]. This abstraction is first described in [Cra96, CD98] and later observed in [Mau15].

Let $(\mathcal{G}, \star)$ and $(\mathcal{H}, \otimes)$ be two groups whose operations are efficiently computable, and let $f : \mathcal{G} \to \mathcal{H}$ be a one-way homomorphism from $\mathcal{G}$ to $\mathcal{H}$. That is, $f(x \star y) = f(x) \otimes f(y)$.

The pre-image protocol Π for relation $\mathcal{R} = \{(x, w) : x = f(w)\}$ with associated algorithm $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{V}))$ and with challenge length l is described below:

- Common input: (description of) $\mathcal{G}$ and $\mathcal{H}$ and $x \in \mathcal{H}$;
- Prover's private input: w such that $x = f(w)$.
- Algorithm P_1.

  On input $(x, w) \in \mathcal{R}$ and random coin tosses R_1, P_1 picks $k \leftarrow \mathcal{G}$, sets $a \leftarrow f(k)$ and outputs a.
- Algorithm P_2.

  On input $(x, w) \in \mathcal{R}$, k, and challenge c, P_2 computes and outputs $z = k \star w^c$.
- Algorithm V.

  On input (x, a, c, z), V outputs 1 iff $f(z) = a \otimes x^c$.

The simulator Sim of Π on input instance x and challenge c works as follows:

- randomly pick $z \leftarrow \mathcal{G}$;
- compute $a = f(z) \otimes x^{-c}$;
- return (a, z).

Theorem 3 of [Mau15] describes the two conditions under which the pre-image protocol is a Σ-protocol. Specifically, for integer y, $u \in \mathcal{G}$ and $(x, w) \in \mathcal{R}$ we have:

- $\gcd(c_1 - c_2, y) = 1$, for all challenges $c_1 \neq c_2 \in \{0, 1\}^l$;
- $f(u) = x^y$.

B.1 Pre-Image Protocol is a Chameleon Σ-Protocol

Theorem 15. *The Pre-Image Protocol is a Chameleon Σ-protocol.*

Proof. We describe algorithm $\mathsf{P}_{\mathsf{sim}}$. Let $(a, \widetilde{z})$ be the output of Sim on input x and challenge $\widetilde{c}$. PPT algorithm $\mathsf{P}_{\mathsf{sim}}$ on input $x, \widetilde{c}$ and the witness w for x and challenge c, computes and outputs $z = \widetilde{z} \star w^{-\widetilde{c}} \star w^c$.

The triple (a, c, z) is an accepting transcript because the test $(f(z) = a \otimes x^c)$ of V is successful. Indeed we have

$$a \otimes x^c = f(\widetilde{z}) \otimes x^{-\widetilde{c}} \otimes x^c = f(\widetilde{z}) \otimes f(w)^{-\widetilde{c}} \otimes f(w)^c = f(\widetilde{z} \star w^{-\widetilde{c}} \star w^c) = f(z).$$

We show now the property of indistinguishability for Chameleon Σ-protocols. We prove that for all pairs of challenges $\widetilde{c}$ and c and for all $(x, w) \in \mathcal{R}$ the two following distributions are indistinguishable:

- (a, c, z), where $a = f(z) \otimes x^{-c}$;
- (a, c, z), where $a = f(\widetilde{z}) \otimes x^{-\widetilde{c}}$ $z = \widetilde{z} \star w^{-\widetilde{c}} \star w^{c}$.

From the SHVZK property follows that the first distribution is perfect indistinguishable from a real transcript $a = f(k)$, c, $z = k \star w^c$ (where k is a random element of $\mathcal{G}$) produced by Π.

We note that $\widetilde{z}$ is a random element of $\mathcal{G}$, so $\widetilde{z} \star w^{-\widetilde{c}}$ is also a random element of $\mathcal{G}$, for this reason we can set $\bar{k} = \widetilde{z} \star w^{-\widetilde{c}}$, and obtain that the second distribution $a = f(\bar{k})$, c, $z = \bar{k} \star w^c$ is identically distributed to a transcript given in output by Π. □

C Classification of Σ-Protocols

In this section, we provide examples for the four classes of Σ-protocols mentioned in Section 1. Table 1 summarizes the classes of Σ-protocols that can be used to construct either a 2-IDTC scheme or a 3-IDTC scheme, or both, and the class of Σ-protocols that cannot be used to instantiate any of our IDTC schemes.

	(Class 1)	*(Class 2)*	*(Class 3)*	*(Class 4)*
2-IDTC	Yes	Yes	No	No
3-IDTC	Yes	No	Yes	No

Table 1: Class of Σ-protocols and their compatibility with IDTC scheme.

Examples of Class 1 and Class 3 Σ-protocols. The Class 1 is the class of Σ-protocols that are Chameleon and that are witness-delayed Σ-protocols. Schnorr's Σ-protocol for DLog is an example of Class 1 Σ-protocol. Indeed, its first round consists of the prover sending a random group element. Moreover, even if this value is computed by a simulator, knowledge of the witness and of the randomness used by the simulator suffices for the prover to answer any challenge.

The Class 3 is the class of Σ-protocols that are not Chameleon and that are witness-delayed Σ-protocols. For instance, Blum's Σ-protocol for Hamiltonian graphs belongs to Class 3. In fact this Σ-protocol requires the prover only to know the graph to compute the first round.

An example of a Class 2 Σ-protocol. Recall that Class 2 is the class of Σ-protocols that are Chameleon and that are not witness-delayed Σ-protocols. We construct a Class 2 Σ-protocol $\Pi = (\mathcal{P}, \mathcal{V})$ for the relation $\mathsf{DLOG} = \{((\mathcal{G}, q, g, Y), y) : g^y = Y\}$ by using Pedersen's commitment scheme [Ped91] as a 2-IDTC scheme. The TCom algorithm of the 2-IDTC scheme for DLOG based on Pedersen's commitment takes as input the description of a cyclic group $\mathcal{G}$ of order q, a generator g of $\mathcal{G}$ and an element $Y \in \mathcal{G}$. To commit to m, TCom selects $r \leftarrow \mathbb{Z}_q$ uniformly at random and returns $\mathtt{com} = g^r \cdot Y^m, \mathtt{dec} = r, \mathtt{rand} = r$. The decommitment algorithm TDec is straightforward and the trapdoor algorithm TFake, knowing the discrete log of Y, can open the commitment com as any message m'.

We are now ready to describe our proposed Class 2 Σ-protocol for DLOG with common input $(\mathcal{G}, q, g, Y)$. In the first round, $\mathcal{P}$ computes $(\mathtt{com}_p, \mathtt{dec}_p^0, \mathtt{rand}) \leftarrow \mathsf{TCom}(\mathcal{G}, g, Y, 0)$ and then uses TFake to compute an opening $\mathtt{dec}_p^1$ of $\mathtt{com}_p$ as 1. Finally, $\mathcal{P}$ computes $(\mathtt{com}_0, \mathtt{dec}_0, \mathtt{rand}_0) \leftarrow \mathsf{TCom}(\mathcal{G}, g, Y, \mathtt{dec}_p^0)$ and $(\mathtt{com}_1, \mathtt{dec}_1, \mathtt{rand}_1) \leftarrow \mathsf{TCom}(\mathcal{G}, g, Y, \mathtt{dec}_p^1)$ and sends $(\mathtt{com}_p, \mathtt{com}_0, \mathtt{com}_1)$ to $\mathcal{V}$ that replies with a one bit challenge b. $\mathcal{P}$ answers by sending $\mathtt{dec}_p^b$ and $\mathtt{dec}_b$. The simulator Sim for the Special HVZK receives an instance $(\mathcal{G}, q, g, Y)$ and a bit b and computes $(\mathtt{com}_p, \mathtt{dec}_p^b, \mathtt{rand}) \leftarrow \mathsf{TCom}((\mathcal{G}, q, g, Y), b)$. Commitment $\mathtt{dec}_p^b$ is committed twice obtaining $\mathtt{com}_0$ and $\mathtt{com}_1$ and only $\mathtt{com}_b$ is opened. Notice that, since Pedersen's commitment is perfectly hiding, then the simulation of Sim is perfect. Clearly, $\mathcal{P}$ needs the witness of Y for computing the first round. Moreover, the proposed protocol is Chameleon since Sim commits twice to the same opening of the commitment $\mathtt{com}_p$ but then $\mathcal{P}$, once the discrete log of Y becomes available, can computed $\mathtt{dec}_p^{1-b}$ and then open $\mathtt{com}_{1-b}$ as $\mathtt{dec}_p^{1-b}$.

Examples of Class 4 Σ-protocols. Recall that Class 4 is the class of Σ-protocols that are not Chameleon and that are not witness-delayed Σ-protocols.

As an example of a Class 4 Σ-protocol we consider the protocol obtained from the Class 2 Σ-protocol described in the previous paragraph in which $\mathtt{dec}_p^0$ and $\mathtt{dec}_p^1$ are committed by using a (non-interactive) commitment scheme that is perfectly binding e computationally hiding instead of a Pedersen's commitment scheme. For example, the ElGamal encryption scheme can be used to construct such a commitment scheme. In this case, the Σ-protocol obtained is only computational HVZK.

D Efficiency

In this section we give a briefly comparison our OR transform and the CDS-OR transform in terms of number of modular exponentiations that they involve.

For this comparison we consider a protocol Π^{OR} that proves the knowledge of one out of two discrete logarithms. Therefore the OR transforms has on input Π_0 and Π_1 both corresponding to Schnorr's Σ-protocol.

We consider two cases:

- 1st case: Π_0 is Schnorr's Σ-protocol for relation $\mathsf{DLOG} = \{((\mathcal{G}', q', g', Y'), y') : g'^{y'} = Y'\}$, where q' is prime and $\mathcal{G}'$ is a group of order q' of the quadratic residues modulo p' s.t. $p' = 2q' + 1$, where $|p'| = 2048$ and $|q'| = 2047$. Π_1 is Schnorr's Σ-protocol for relation $\mathsf{DLOG} = \{((\mathcal{G}, q, g, Y), y) : g^y = Y\}$, where q is prime and G is a group of order q of the quadratic residues modulo p s.t. $p = 2q + 1$, where $|p| = 1024$ and $|q| = 1023$.
- 2nd case: Π_0 and Π_1 are both like Π_1 described in the 1st case.

In both cases we instantiate t$-\mathsf{IDTC}$ from Π_0.

The execution of Schnorr's Σ-protocol costs 1 modular exponentiation, while the execution of the simulator of Schnorr's Σ-protocol costs 2 modular exponentiations. By exponentiation mod p or exponentiation mod p', we indicate, respectively the exponentiation modulo a prime of 1024 bits and the exponentiation modulo a prime of 2048 bits.

To evaluate the cost of our OR transform, we first note that the number of modular exponentiations of our OR transform are different when it is instantiated using a $2-\mathsf{IDTC}$ scheme or

3$-$IDTC scheme, even when the 2$-$IDTC scheme and 3$-$IDTC schemes are constructed from the same Σ-protocol. In particular if the scheme are constructed from Schnorr's Σ-protocol run the algorithm TCom costs 2 modular exponentiations in the case of 2$-$IDTC and it costs 3 modular exponentiations in the case of a 3$-$IDTC.

From this observation follows that to compute the first round of our our OR transform we need 1 exponentiation mod p to compute the first round of Schnorr's Σ-protocol plus the cost to execute TCom.

To compute the third round of our OR transform we need 2 exponentiations mod p when we run equivocal procedure TFake, because we need to execute again a simulator of Π_1. Otherwise no other exponentiations is required. Therefore in the worst case to compute the third round of our OR transform we need 2 exponentiations mod p.

Summing up:

- in the 1st case our OR transform costs 3 exponentiations mod p plus 2 exponentiations mod p' if we use a 2$-$IDTC scheme (or 3 exponentiations mod p' if we use a 3$-$IDTC scheme).
- in the 2nd case our OR transform costs 3 exponentiations mod p plus 4 exponentiations mod p if we use a 2$-$IDTC scheme (or 6 exponentiations mod p if we use a 3$-$IDTC scheme). Note that in this case to commit to a first round of Schnorr's Σ-protocol where $|a| = 1024$ bits, we need to run twice the TCom procedure.

The CDS-OR transform costs in both cases 3 modular exponentiations. In the 1st case the 2 exponentiations are mod p and 1 exponentiation is mod p', in the 2nd case all the exponentiations are mod p.