Multi-Theorem Fiat-Shamir Transform from Correlation-Intractable Hash Functions.

Multi-Theorem Fiat-Shamir Transform from Correlation-Intractable Hash Functions

Michele Ciampi and Yu Xia

The University of Edinburgh, UK
{michele.ciampi, yu.xia}@ed.ac.uk

Abstract. In STOC 2019 Canetti et al. showed how to soundly instantiate the Fiat-Shamir transform assuming that prover and verifier have access to the key of a *correlation intractable hash function for efficiently searchable relations*. The transform requires the starting protocol to be a special 3-round public-coin scheme that Canetti et al. call *trapdoor sigma-protocol*. One downside of the Canetti et al. approach is that the key of the hash function can be used only once (or a pre-determined bounded number of times). That is, each new zero-knowledge proof requires a freshly generated hash key (i.e., a freshly generated setup). This is in contrast to what happens with the standard Fiat-Shamir transform, where the prover, having access to the same hash function(modeled as a random-oracle), can generate an unbounded number of proofs that are guaranteed to be zero-knowledge and sound.
As our main contribution, we extend the results of Canetti et al., by proposing a multi-theorem protocol that follows the Fiat-Shamir paradigm and relies on correlation intractable hash functions. Moreover, our protocol remains zero-knowledge and sound even against adversaries that choose the statement to be proven (and the witness for the case of zero-knowledge) adaptively on the key of the hash function. Our construction is presented in the form of a compiler, that follows the Fiat-Shamir paradigm, which takes as input any trapdoor sigma-protocol for the NP-language L and turns it into a non-interactive zero-knowledge protocol that satisfies the properties we mentioned. To be best of our knowledge, ours is the first compiler that follows the Fiat-Shamir paradigm to obtain a multi-theorem adaptive NIZK relying on correlation intractable hash functions.

Keywords: NIZK · Fiat-Shamir Transform · Adaptive Multi-Theorem Zero-Knowledge · Correlation Intractable Hash Functions

Table of Contents

1 Introduction

Non-interactive zero-knowledge (NIZK) proofs [BFM88, DMP88] allow a prover to convince a verifier about the validity of an NP-statement with just one round of interaction (one message that goes from the prover to the verifier). One of the most famous techniques used to realize non-interactive proofs is the *Fiat-Shamir (FS) transform* [FS87]. This transform takes as input a *sigma-protocol* and turns it into a NIZK proof. A sigma-protocol is a special three-round public-coin interactive proof executed between a prover P and a verifier V, where P's goal is to convince V that a common statement x belongs to a given NP language L. The prover knows a witness w (corresponding to x) and starts the interaction by sending a first message a; the verifier then sends a uniformly random bit-string c, called the challenge, to which the prover replies with the last message z. Finally, the verifier decides whether $x \in L$ or not based on x and the transcript (a, c, z).

The FS transform makes a sigma-protocol non-interactive by letting the prover do the sampling of the challenge. In particular, the prover computes $c \leftarrow \mathsf{H}(a)$, where H is a hash function. One way to argue about the security of this construction is by modeling H as a Random Oracle [BR93, FKMV12]. Recently, [BKM20, CCH+19, CCR16, CCRR18, HL18, KRR17, PS19] showed that if the hash function is correlation-intractable (CI) for certain relations, then the resulting NIZK is sound. Informally, the CI property ensures that given a random hash key k, it is computationally difficult to find any input α, s.t. $(\alpha, \mathsf{H}_k(\alpha)) \in R$ for a particular relation R.

In more detail, Canetti et al. [CCH+19] shows that the FS transform remains secure assuming that the hash function is correlation intractable for *efficiently searchable relations*[1]. The result of [CCH+19] can be applied only to a restricted class of sigma-protocols called *trapdoor sigma-protocol*. Trapdoor sigma-protocols are three-round public-coin protocols defined in the Common Reference String (CRS) model that enjoy three main properties: honest verifier zero-knowledge (HVZK), optimal soundness, and admit a *bad-challenge* extractor. The property of HVZK is quite standard and guarantees the existence of a simulator that, upon receiving the challenge (the second round), it produces a transcript that is indistinguishable from the transcript generated via the interaction of an honest prover and verifier. Optimal soundness guarantees that for any statement $x \notin L$ and the first-round message a there exists at most one challenge c, such that a verifier would accept the transcript (a, c, z), for the statement x, for some third-round z. We refer to the unique challenge c as the *bad-challenge*. Finally, the bad-challenge extractor is an algorithm that takes as input a false statement x, a valid first-round a, and some trapdoor information τ, and efficiently computes the bad-challenge c.

Adaptive multi-theorem NIZK. The most basic notion of soundness for a non-interactive proof system guarantees soundness in the presence of an adversary that decides the statement to be proven before the sampling of the CRS. Similarly, the notion of zero-knowledge is guaranteed to hold for any choice of theorem-witness sampled by the adversary non-adaptively on the CRS. We refer to this class of adversaries as *non-adaptive adversaries*. It is possible to consider stronger (and more realistic) notions of security that guarantee that both the soundness and the zero-knowledge hold even if the adversary can make the choice of the theorem to be proven (and of the witness for the zero-knowledge experiment) adaptively on the CRS. In [CCH+19] the authors argue that if the trapdoor sigma-protocol admits a special bad-challenge extractor, and moreover it is *adaptive* special-honest verifier zero-knowledge[2], then the NIZK they obtain using CI hash functions is also adaptive secure. Unfortunately, the only trapdoor sigma-protocol known to satisfy all the required properties is the Lapidot-Shamir [LS91] protocol for Hamiltonian graphs. In [CPV20] the authors show that all sigma-protocols can be turned into trapdoor sigma-protocols with an adaptive HVZK simulator. One drawback of all the previous approaches is that the zero-knowledge property is not preserved if the same hash key is used to generate more than one proof. However, we would like a prover to be able to use the same hash key to generate multiple proofs (for potentially different theorems). We refer to this notion of zero-knowledge as *multi-theorem NIZK*[3], and we investigate the following question:

[1] A relation is efficiently searchable if given x it is efficient to find y such that $(x, y) \in R$

[2] The notion of adaptive HVZK guarantees the existence of a simulator that can generate the first-round of the protocol without the knowledge of the theorem.

[3] The notion we consider in this paper is with respect to a single prover. This single entity can use the same CRS to generate multiple proofs for potentially different statements.

Is it possible to obtain an adaptive multi-theorem NIZK by applying the Fiat-Shamir paradigm using a hash function that is correlation intractable for efficiently searchable relations?

Another way to phrase the above is that we ask whether it is possible to construct an adaptive multi-theorem NIZK using the same setup (and complexity) assumption as in [CCH+19, CPV20].

1.1 Our results

In this work we show how to obtain an *adaptive multi-theorem* NIZK for any language L that admits a trapdoor sigma-protocol Σ_L (we do not require Σ_L to be adaptive HVZK). The nice feature of our NIZK is that the prover, after a pre-processing (non-interactive) phase, upon receiving the statement to be proven and the corresponding witness, generates proofs by just following the FS paradigm.

Due to its FS-like structure, the soundness of our scheme relies only on the security of the underlying trapdoor sigma-protocols and on the correlation-intractability of the hash function (exactly as in all previous works that although achieved a weaker form of zero-knowledge). The zero-knowledge property instead relies on the HVZK of the trapdoor sigma-protocols, the security of the CI hash function, and the hardness of the Decisional Diffie-Hellman (DDH) assumption. This is exactly in the same spirit as [CCH+19, CPV20] where the authors instead rely on the hardness of public-key encryption schemes to argue about zero-knowledge. An informal theorem that summarizes our result is the following

Theorem (informal): *If Σ_L is a trapdoor sigma-protocol for the language L, then it is possible to realize an* adaptive multi-theorem *NIZK protocol that follows the FS paradigm. In particular, the soundness of the NIZK protocol depends only on the soundness of underlying trapdoor sigma-protocols and on the security of the hash function.*

We note that an easy way to construct a multi-theorem NIZK would be to use the OR approach proposed in [FLS90]. In this, a statement $T \notin L^\star$ for a membership-hard language[4] $L^\star$ is put in the CRS, and the prover provides an OR proof proving that either $x \in L$ or $T \in L^\star$. This approach has two main drawbacks: 1) the NIZK is inherently computational zero-knowledge and 2) the soundness holds only under the condition that the tuple T is sampled such that $T \notin L^\star$. In our work, we show how to modify the FLS approach to remove the second limitation. Hence, we obtain a NIZK that has exactly the same setup assumptions as previous works, but in addition, we obtain a protocol that is multi-theorem.

[4] Intuitively, a membership-hard language is one for which it is possible to sample instances of the problem in a way that it is hard to detect if a given instance is in the language or not

1.2 Technical overview

Adaptive multi-theorem NIZK from CI hash functions. We first recall the approach proposed in [FLS90] used to realize an adaptive multi-theorem NIZK protocol for an NP language L. In this, the prover generates an OR proof showing that either $x \in L$ or that $T \in L^\star$, where T is an instance that is part of the CRS. The soundness holds due to the soundness of the OR proof and the fact that by the construction of the CRS $T \notin L^\star$. The adaptive zero-knowledge comes from the fact that a simulator, to generate simulated proofs needs to program the CRS with $T^\star \in L^\star$ (and for this no knowledge about the statement to be proven is needed). Upon receiving a statement x, the simulator uses the witness for $T^\star$ to generate the OR proof. If the OR proof is witness-indistinguishable (WI), and $L^\star$ is a membership-hard language, then the protocol is adaptive zero-knowledge. The multi-theorem feature comes from the fact that the WI property is closed under sequential composition.

By relying on the result of [CDS94], it is possible to compile two sigma-protocols, respectively for the language L_1 and L_2, into a new sigma-protocol for the OR language $L_1 \vee L_2$. In this paper, we argue that the compiler of [CDS94] works similarly for trapdoor sigma-protocols. This means that if we have a trapdoor sigma-protocol for L and one for $L^\star$, we can obtain an adaptive multi-theorem NIZK protocol by doing the following. First, we obtain a trapdoor sigma-protocol for the language $L \vee L^\star$, and then we apply the FS transform to the resulting protocol thus obtaining a NIZK protocol for the language $L \vee L^\star$.

The scheme we have just described departs from the FS paradigm mostly due to the presence of the T value embedded in the CRS (that the simulator needs to program as we have discussed earlier). In the FS paradigms, such a component is not required, since the simulator only needs to program the hash function to perform the final simulation. But more importantly, the value T needs to be correctly generated, (i.e., it must not belong to $L^\star$ otherwise the soundness does not hold). This is clearly something undesirable since now the soundness does not only rely on the security of the hash function (which is the case for the FS transform) but also requires additional parameters to be generated honestly.

We work around this problem as follows. We define $L^\star$ as being the language of all the DH tuples, and instead of requiring the CRS to contain $T \notin L^\star$, we let the prover pick the tuple T. We then require the prover to provide a non-interactive zero-knowledge proof via a protocol Π_{NDH} thus proving that the tuple does not belong to $L^\star$ (i.e., T is non-DH). Note that we require Π_{NDH} to be a NIZK protocol that guarantees security only if one proof is generated (i.e., it is not multi-theorem zero-knowledge). In particular, Π_{NDH} can be instantiated via the Fiat-Shamir transform using a correlation intractable hash function on a specific trapdoor sigma-protocol (we will elaborate more on this in the technical part of the paper). The rest of the protocol follows as before. That is, the prover, upon receiving a statement x and its witness, perform an OR proof, proving either that $x \in L$ or that T is a DH tuple.

The main observation here is that Π_{NDH} needs to be run only once, and the obtained proof can be reused any time the prover is required to generate a proof

for a new instance x. So, we can see our protocol as divided into two phases. In the *offline phase* the prover samples a non-DH tuple T, and runs Π_{NDH} to generate a NIZK proof that we denote with π^{NDH} (without sending it). Upon receiving a statement and a witness, the prover generates the OR proof π^{OR}, and sends over $(\pi^{\mathsf{OR}}, T, \pi^{\mathsf{NDH}})$.

We prove that the protocol we have just described is adaptive multi-theorem zero-knowledge. Intuitively, this holds since the simulator can fake the proof for the non-DH tuple by running the simulator of Π_{NDH}. Then the proof π^{NDH} can be simulated with respect to a DH tuple, hence any OR proof can be generated using the fact that $T \in L^{\star}$. Given that the OR proof we will use is witness indistinguishable (WI), and that the WI property is maintained under parallel composition, then our final protocol is multi-theorem zero-knowledge. The adaptive zero-knowledge property comes from the fact that the simulator can run internally the simulator of Π_{NDH} to generate the setup (i.e., to program the hash function) without knowing x.

There is a caveat about this protocol. Note that the tuple T can be chosen by the adversarial prover adaptively on the description of the hash function. So, even if we do not need Π_{NDH} to be multi-theorem, it seems that we need it to be at least *adaptive-sound*. To obtain an adaptive-sound NIZK protocol following the FS paradigm, we could rely on the results of [CPV20]. In this, the authors show how to convert any sigma-protocol into an adaptive-sound NIZK protocol using correlation intractable hash functions. However, the Ciampi et al. compiler incurs an efficiency loss, since it requires, for each bit of the challenge of the starting sigma-protocol, to generate two ciphertexts. To avoid this, we first argue that it is sufficient to fix the first two components of the tuple T (g, g^{α}) in the CRS, and let the adversarial prover choose only X, Y adaptively on the hash function to form the tuple $T = (g, g^{\alpha}, X, Y)$. We then show how to obtain a protocol Π_{NDH} that remains sound in this *semi-adaptive* adversarial setting, while maintaining reasonable performance (i.e., for a security parameter of 1024 bits the prover and verifier of Π_{NDH} need to perform 40 exponentiations each).

We need to argue that the OR proof also remains sound when part of the tuple T is chosen by the adversary. In the technical part of the paper, we will show how to realize such an OR proof and provide our new formal definition of soundness that we call *semi-adaptive soundness*, which allows the adversary to decide part of the component of an NP statement. This notion lies in between the standard notion of soundness and the notion of adaptive soundness, which allows the adversary to decide all the parameters of the NP instance to be proven.

On adaptive soundness. So far we have mostly focused on obtaining an adaptive zero-knowledge scheme that allows the re-use of the hash-key. We have not mentioned whether it is possible to also prove that our NIZK is adaptive sound. We argue that if the trapdoor sigma-protocol Π_L admits a special type of extractor (in [CPV20] the authors show that any sigma-protocol can be modified to enjoy this special property), then our NIZK is also adaptive-sound. We refer to the technical part of our paper for more detail.

1.3 Related work

One of the works most related to ours is [CSW20]. In this, the authors construct an adaptive sound, adaptive zero-knowledge, multi-theorem NIZK from correlation intractable hash functions (plus other assumptions like LWEs, or DDH and LPN). However, the results of [CSW20] follow a different spirit compared to ours (and compared also to [CCH+19]). As discussed in the previous section, a multi-theorem adaptive NIZK can be trivially obtained using a folklore technique. Namely, it is easy to construct an adaptive multi-theorem NIZK protocol from the same assumptions we use in our paper by following the FLS approach. However, this approach produces a CRS that has two components: a hash key, and a tuple $T \notin L^{\star}$. Hence, the soundness of the protocol depends on T not being in $L^{\star}$. This is in contrast with what happens in the standard FS transform where the soundness depends only on the soundness of the underlying sigma protocol and on the CI of the hash function. All the multi-theorem protocols proposed in [CSW20] have a similar drawback. That is, the soundness is based on a public key (that is part of the CRS) being sampled correctly. If such a public key is not sampled correctly then the soundness trivially does not hold. In our work, we instead get the same advantage of the FS approach (and of the results proposed in [CCH+19]) by providing a protocol whose soundness is based on the correlation intractability of the hash function and on the soundness of the underlying trapdoor sigma-protocol only. To give a concrete example of the benefit of our compiler compared to existing solutions we note the following. If we instantiate our NIZK with the trapdoor sigma-protocol for the language of Diffie-Hellman tuples, we obtain a multi-theorem adaptive NIZK where the CRS consists of the hash key, and two group elements (g, h). The soundness of this NIZK then holds as long as the hash-key is honestly generated, while (g, h) can be maliciously generated.

Our work follows the spirit of [CCH+19], where the authors show how to compile any trapdoor sigma protocol into a NIZK using the FS approach. We extend the approach of Canetti et al. proposing a compiler that turns *any* trapdoor sigma-protocol into a *multi-theorem adaptive* NIZK. Hence, any improvement in the efficiency of trapdoor sigma protocols has an immediate impact on the performance of our NIZK. [CSW20] follows a different path by proposing ad-hoc schemes that depart from the Fiat-Shamir approach. The advantage of [CSW20] over our work is that the results of [CSW20] are UC secure and tolerate adaptive corruption.

2 Preliminaries

Notations. We denote the security parameter by λ and use "||" as the concatenation operator. For a finite set Q, $x \leftarrow\!\!\$\, Q$ denotes a sampling of x from Q with uniform distribution. We use " $=$ " to check the equality of two different elements, " $\leftarrow$ " as the assigning operator (e.g. to assign to a the value of b we write $a \leftarrow b$). We use the abbreviation PPT which stands for probabilistic polynomial time. We use $\mathsf{poly}(\cdot)$ to indicate a generic polynomial function. We

denote with $\mathbb{Z}_p$ the set of integers, where p is the order of the set, with $\mathbb{N}$ the set of natural numbers . We use $G.\mathsf{Gen}(1^\lambda)$ to represent the algorithm to find the generator in the group G. ν represents the negligible function, and δ represents the non-negligible function. For an NP language L we denote the corresponding NP-relation with $\mathcal{R}_L$.

We assume familiarity with the notions of negligible and non-negligible functions, and also the notion of interactive proof systems.

2.1 Diffie-Hellman related definitions

Let G be the group of an order p, with a generator g. Let $T = (g, h = g^x, X, Y)$ be a tuple, where $x \in \mathbb{Z}_p$. Let $L_{\mathsf{DH}} = \{T \in G^4 \mid \exists w \in \mathbb{Z}_p : X = g^w \wedge Y = h^w\}$ be the language of DH tuples. Let $L_{\mathsf{NDH}} = \{T \in G^4 \mid \exists w, w' \in \mathbb{Z}_p : X = g^w \wedge Y = h^{w'} \wedge w \neq w'\}$ be the language of non-DH tuples.

We assume the Decisional Diffie-Hellman (DDH) hardness assumption holds in the group G. The DDH hardness assumption is as follows:

Definition 1 (DDH hardness Assumption). *For every* PPT *algorithm* $\mathcal{A}$*:*

$$\Big|\Pr[\mathcal{A}(T) = 1 \mid T \in L_{\mathsf{DH}}] - \Pr[\mathcal{A}(T) = 1 \mid T \in L_{\mathsf{NDH}}]\Big| \leq \nu(\lambda).$$

2.2 Non-Interactive Argument Systems related definitions

We recall the notion of non-interactive argument systems here.

Definition 2 (Non-Interactive Zero-Knowledge Argument Systems). *A* non-interactive zero-knowledge argument system (NIZK) *for an NP-language L with the corresponding relation R_L is a non-interactive protocol $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$, where:*

- $\mathsf{Setup}(1^n, 1^\lambda)$ *takes as the input a statement length n and a security parameter λ. It outputs a common reference string crs.*
- $\mathsf{P}(crs, x, w)$ *takes as the input crs, the statement x and the witness w, s.t. $(x, w) \in R_L$. It outputs the proof π.*
- $\mathsf{V}(crs, x, \pi)$ *takes as the input crs, x and π. It outputs 1 to accept and 0 to reject.*

Π has the following properties:

- **Completeness.** *For all $\lambda \in \mathbb{N}$, and all $(x, w) \in R_L$, it holds that:*

$$\Pr\Big[\mathsf{V}(crs, x, \mathsf{P}(crs, x, w)) = 1 \;\Big|\; crs \leftarrow\!\$\; \mathsf{Setup}(1^{|x|}, 1^\lambda)\Big] = 1 - \nu(\lambda)$$

- **Soundness.** *For all* PPT *provers $\mathsf{P}^\star$, s.t. for all $\lambda \in \mathbb{N}$, and all $x \notin L$, it holds that:*

$$\Pr\Big[\mathsf{V}(crs, x, \pi) = 1 \;\Big|\; crs \leftarrow\!\$\; \mathsf{Setup}(1^{|x|}, 1^\lambda); \pi \leftarrow\!\$\; \mathsf{P}^\star(crs)\Big] \leq \nu(\lambda).$$

- **Zero knowledge.** *There exists a* PPT *simulator* Sim *such that for every $(x, w) \in R_L$, the distribution ensembles $\{(crs, \pi) : crs \leftarrow\!\$\; \mathsf{Setup}(1^{|x|}, 1^\lambda); \pi \leftarrow\!\$\; \mathsf{P}(crs, x, w)\}_{\lambda \in \mathbb{N}}$ and $\{\mathsf{Sim}(1^\lambda, x)\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable.*

2.3 Sigma-protocol related definitions

Most of the following definitions are taken from [CCH+19, CPSV16].

Definition 3 (Sigma-protocol). *Assuming there is a three-round public-coin interactive protocol $\Sigma = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ for a NP language L (and corresponding relation R_L) in the common reference string model, where:*

- Gen *takes as input the unary representation of the security parameter, and it outputs the common reference string crs.*
- *In the first round of the protocol, P takes as input the common reference string crs, the instance x, the witness w, the randomness R, and it will output the first round message a.*
- *In the second round, V takes as input the crs, x, a, and it will output the challenge c.*
- *In the third round, P takes as input the crs, x, w, a, c, R, and it will output the third round message z.*
- *When V receives (crs, x, a, c, z) as inputs, it outputs 1 to accept and 0 to reject.*

Σ is a sigma-protocol if satisfies the following properties:

- ***Completeness**: If $(x, w) \in R_L$, then all honest generated transcripts are accepting.*
- ***Optimal soundness**: For every common reference string crs, every instance $x \notin L$, and every first message a, there is at most one challenge $c = f(crs, x, a)$ such that (crs, x, a, c, z) is an accepting transcript for any choice of third message z. We informally call f the "bad-challenge function" associated with Σ and note that f may not be efficiently computable.*
- ***Special HVZK**: There exists a PPT simulator algorithm Sim that takes as $x \in L$ and $c \in \{0,1\}^\ell$, and outputs an accepting transcript for x where c is the challenge (we denote this action with $(a, z) \leftarrow \mathsf{Sim}(x, c)$). 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 honest generated transcript obtained when V sends c as the challenge and P runs on common input x and any private input w such that $(x, w) \in R_L$.*

Remark 1. The definition 3 is a bit different from the standard notion of sigma-protocol [Dam10] since we only require the protocol to be the optimal sound (instead of special-sound).

Then we recall the definition of the instance-dependant trapdoor sigma-protocol from [CCH+19].

Definition 4 (Instance-dependant trapdoor sigma-protocol [CCH+19]). *We say that a sigma-protocol $\Sigma = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ with bad-challenge function f is an instance-dependant trapdoor sigma-protocol if there are PPT algorithms $\mathsf{TrapGen}$, $\mathsf{BadChallenge}$ with the following syntax.*

– $\mathsf{TrapGen}(1^\lambda, x, aux)$ *takes as input the unary representation of the security parameter, an instance* x*, and an auxiliary input* aux*. It outputs a common reference string* crs *along with a trapdoor* τ*.*
– $\mathsf{BadChallenge}(\tau, crs, x, a)$ *takes as input a trapdoor* τ*, common reference string* crs*, instance* x*, and first message* a*. It outputs a challenge* c*.*

We additionally require the following properties:

– ***CRS Indistinguishability****: For any* (x, aux)*, an honestly generated common reference string* crs *is computationally indistinguishable from a common reference string output by* $\mathsf{TrapGen}(1^\lambda, x, aux)$*.*
– ***Correctness****: For every instance* $x \notin L$*, there exists an auxiliary input* aux *such that for all* $(crs, \tau) \leftarrow\!\$\ \mathsf{TrapGen}(1^\lambda, x, aux)$*, we have that* $\mathsf{BadChallenge}(\tau, crs, x, a) = f(crs, x, a)$*.*

OR composition of ID trapdoor sigma-protocols. In our paper, we also argue that the OR composition [CDS94] of any 2 instance-dependant trapdoor sigma-protocols (for the relation R_{L_0} and R_{L_1}) is an instance-dependant trapdoor sigma-protocol for the relation $R_{L_0 \vee L_1}$. Moreover, the resulting protocol is witness indistinguishable (WI).

We recall the OR composition techniques here in Fig. 1. Assuming we have 2 three-round public-coin HVZK proof systems $\Sigma_{L_0} = (\mathsf{Gen}_{L_0}, \mathsf{P}_{L_0}, \mathsf{V}_{L_0})$ for NP language L_0 (The corresponding relation is R_{L_0}), $\Sigma_{L_1} = (\mathsf{Gen}_{L_1}, \mathsf{P}_{L_1}, \mathsf{V}_{L_1})$ for NP language L_1 (The corresponding relation is R_{L_1}). Then the three-round public-coin HVZK proof system $\Sigma_{L_0 \vee L_1} = (\mathsf{Gen}_{L_0 \vee L_1}, \mathsf{P}_{L_0 \vee L_1}, \mathsf{V}_{L_0 \vee L_1})$ is for the NP language $L_{L_0 \vee L_1}$ defined below (The corresponding relation is $R_{L_0 \vee L_1}$):

$$L_{L_0 \vee L_1} = \{(x_0, x_1) : x_0 \in L_0 \vee x_1 \in L_1\}$$

The $\mathsf{Gen}_{L_0 \vee L_1}$ algorithm works as follows:

– $crs_{L_0} \leftarrow\!\$\ \mathsf{Gen}_{L_0}(1^\lambda)$, $crs_{L_1} \leftarrow\!\$\ \mathsf{Gen}_{L_1}(1^\lambda)$
– output (crs_{L_0}, crs_{L_1})

Then, we let the challenge space be $\{0,1\}^\lambda$, let $b \in \{0,1\}$, let $\mathsf{Sim}_{L_{1-b}}$ be the simulator for $\Sigma_{L_{1-b}}$, and we let w be the witness for instance x_b. In other words, $(x_b, w) \in R_{L_b}$. The protocol $\Sigma_{L_0 \vee L_1}$ is in Fig. 1, in the CRS model, where $crs_{L_0 \vee L_1} \leftarrow\!\$\ \mathsf{Gen}_{L_0 \vee L_1}(1^\lambda)$:

Then we first prove the following Lemma 1.

Lemma 1. *$\Sigma_{L_0 \vee L_1}$ has an efficient bad-challenge extractor $\mathsf{BadChallenge}_{L_0 \vee L_1}$, and the correctness property holds.*

Proof. For $\Sigma_{L_0 \vee L_1}$, $x \notin L_{L_0 \vee L_1}$ means, for the statement $x = (x_0, x_1)$, s.t. $x_0 \notin L_0 \wedge x_1 \notin L_1$. Based on construction of $\Sigma_{L_0 \vee L_1}$ (shown in Figure 1), we have the first round message $a = (a_0, a_1)$, where a_0 is for Σ_{L_0} and a_1 is for Σ_{L_1}.

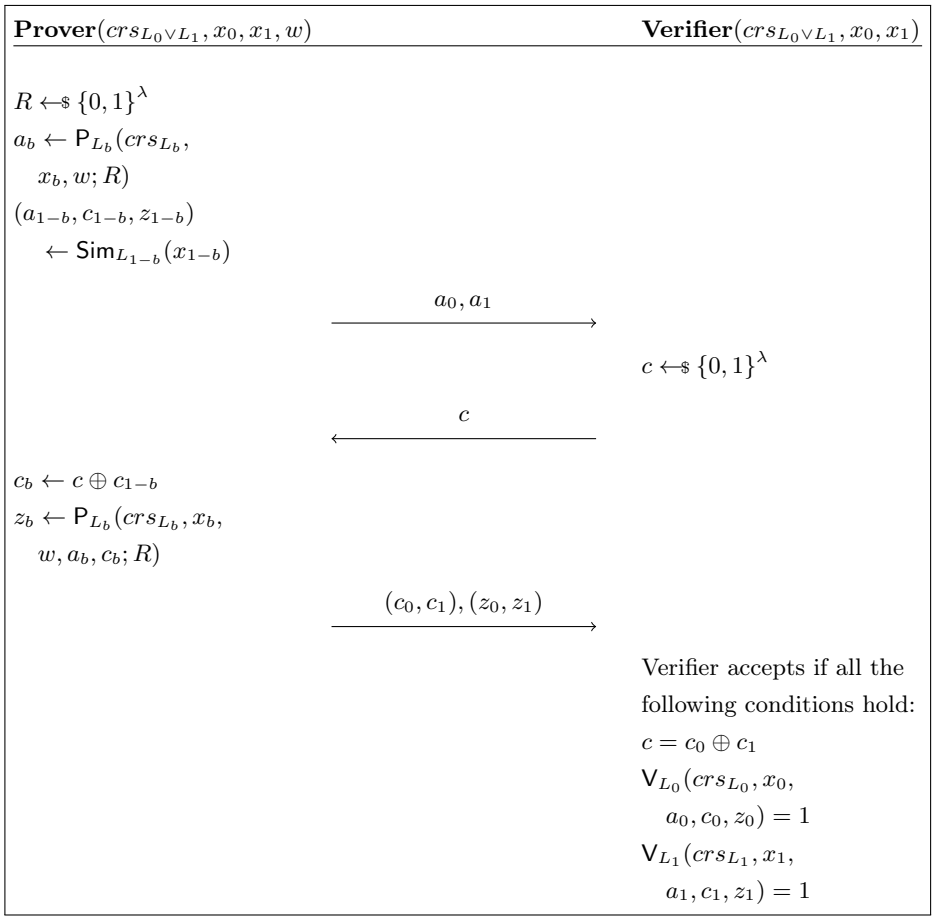

Fig. 1. The protocol for OR composition

Given $aux_{L_0 \vee L_1} = (aux_{L_0}, aux_{L_1})$, the $\mathsf{TrapGen}_{L_0 \vee L_1}$ algorithm is:

$\mathsf{TrapGen}_{L_0 \vee L_1}(1^\lambda, x, aux_{L_0 \vee L_1})$:
$(crs_{L_0}, \tau_{L_0}) \leftarrow\!\$ \mathsf{TrapGen}_{L_0}(1^\lambda, x_0, aux_{L_0})$
$(crs_{L_1}, \tau_{L_1}) \leftarrow\!\$ \mathsf{TrapGen}_{L_1}(1^\lambda, x_1, aux_{L_1})$
$crs_{L_0 \vee L_1} \leftarrow (crs_{L_0}, crs_{L_1})$
$\tau_{L_0 \vee L_1} \leftarrow (\tau_{L_0}, \tau_{L_1})$
return $(crs_{L_0 \vee L_1}, \tau_{L_0 \vee L_1})$

With the output of $\mathsf{TrapGen}_{L_0 \vee L_1}$, the bad-challenge extractor $\mathsf{BadChallenge}_{L_0 \vee L_1}$ works as follows:

$$\begin{aligned}
&\mathsf{BadChallenge}_{L_0 \vee L_1}(\tau_{L_0 \vee L_1}, crs_{L_0 \vee L_1}, x, a): \\
&\quad c_0 \leftarrow \mathsf{BadChallenge}_{L_0}(\tau_{L_0}, crs_{L_0}, x_0, a_0) \\
&\quad c_1 \leftarrow \mathsf{BadChallenge}_{L_1}(\tau_{L_1}, crs_{L_1}, x_1, a_1) \\
&\quad c = c_0 \oplus c_1 \\
&\quad \text{return } c
\end{aligned}$$

The Correctness of $\Sigma_{L_0 \vee L_1}$ is proven by the following reduction:

– Assuming the Correctness property of $\Sigma_{L_0 \vee L_1}$ does not hold. It means there exists $a = (a_0, a_1)$ for all $x = (x_0, x_1), aux_{L_0 \vee L_1} = (aux_{L_0}, aux_{L_1})$, s.t.:

$$\begin{aligned}
&\mathsf{BadChallenge}_{L_0 \vee L_1}(\tau_{L_0 \vee L_1}, crs_{L_0 \vee L_1}, x, a) \neq f(crs_{L_0 \vee L_1}, x, a) \mid \\
&(crs_{L_0 \vee L_1}, \tau_{L_0 \vee L_1}) \leftarrow\!\$\ \mathsf{TrapGen}_{L_0 \vee L_1}(1^\lambda, x, aux_{L_0 \vee L_1})
\end{aligned}$$

Based on the construction of $\mathsf{BadChallenge}_{L_0 \vee L_1}$, we know that Correctness of $\Sigma_{L_0 \vee L_1}$ does not hold is because

$$\begin{aligned}
&\mathsf{BadChallenge}_{L_0}(\tau_{L_0}, crs_{L_0}, x_0, a_0) \neq f(crs_{L_0}, x_0, a_0) \\
&\quad \vee\ \mathsf{BadChallenge}_{L_1}(\tau_{L_1}, crs_{L_1}, x_1, a_1) \neq f(crs_{L_1}, x_1, a_1)
\end{aligned}$$

Then we can discuss the following situations:

- Assuming $\mathsf{BadChallenge}_{L_0}(\tau_{L_0}, crs_{L_0}, x_0, a_0) = f(crs_{L_0}, x_0, a_0)$, then we can construct an adversary $\mathcal{A}'$ for breaking the Correctness of Σ_{L_1}:

$$\begin{aligned}
&\mathcal{A}'(\tau_{L_1}, crs_{L_1}): \\
&\quad (crs_{L_0 \vee L_1}, \tau_{L_0 \vee L_1}) \leftarrow\!\$\ \mathsf{TrapGen}_{L_0 \vee L_1}(1^\lambda, x, aux_{L_0 \vee L_1}) \\
&\quad \text{parsing } crs_{L_0 \vee L_1} \text{ as } (crs_0, crs_1),\ \text{parsing } \tau_{L_0 \vee L_1} \text{ as } (\tau_0, \tau_1) \\
&\quad crs_{L_0 \vee L_1} \leftarrow (crs_0, crs_{L_1}), \tau_{L_0 \vee L_1} \leftarrow (\tau_0, \tau_{L_1}) \\
&\quad a \leftarrow \mathcal{A}(\tau_{L_0 \vee L_1}, crs_{L_0 \vee L_1}) \\
&\quad \text{parsing } a \text{ as } (a_0, a_1) \\
&\quad \text{return } a_1
\end{aligned}$$

  the output of $\mathcal{A}'$ finds a_1 makes

$$\begin{aligned}
&\mathsf{BadChallenge}_{L_1}(\tau_{L_1}, crs_{L_1}, x_1, a_1) \neq f(crs_{L_1}, x_1, a_1) \mid \\
&(crs_{L_1}, \tau_{L_1}) \leftarrow\!\$\ \mathsf{TrapGen}_{L_1}(1^\lambda, x_1, aux_{L_1})
\end{aligned}$$

  for all $x_1 \notin L_1$ and aux_{L_1}. It contradicts to the Correctness of Σ_{L_1}.
- We can do a similar reduction to the Correctness of Σ_{L_0} if the Correctness of Σ_{L_1} holds.
- If $\mathsf{BadChallenge}_{L_0}(\tau_{L_0}, crs_{L_0}, x_0, a_0) \neq f(crs_{L_0}, x_0, a_0) \wedge \mathsf{BadChallenge}_{L_1}(\tau_{L_1}, crs_{L_1}, x_1, a_1) \neq f(crs_{L_1}, x_1, a_1)$, then we can still use $\mathcal{A}'$ to break Correctness of Σ_{L_1}.

It is also important to note that, $\mathsf{BadChallenge}_{L_0 \vee L_1}$ is efficient, because $\mathsf{BadChallenge}_{L_0}$, $\mathsf{BadChallenge}_{L_1}$, and the $\oplus$ operation are efficient. □

Then we have the following Lemma 2:

Lemma 2. *$\Sigma_{L_0 \vee L_1}$ is an instance-dependant trapdoor sigma-protocol and it is witness indistinguishable for the language $L = \{(x_0, x_1) : x_0 \in L_0 \vee x_1 \in L_1\}$*

Proof. By Lemma 1, we know that $\Sigma_{L_0 \vee L_1}$ has $\mathsf{TrapGen}$ and $\mathsf{BadChallenge}$ algorithms as required in the definition. Then we need to prove CRS Indistinguishability and Correctness:

- CRS Indistinguishability: It is important to note that the honested generated CRS for $\Sigma_{L_0 \vee L_1}$ is $crs^{\mathsf{Real}}_{L_0 \vee L_1} = (crs^{\mathsf{Real}}_{L_0}, crs^{\mathsf{Real}}_{L_1})$, where $crs^{\mathsf{Real}}_{L_0} \leftarrow\!\!\$\ \mathsf{Gen}_{L_0}(1^\lambda)$, and
  $crs^{Real}_{L_1} \leftarrow\!\!\$\ \mathsf{Gen}_{L_1}(1^\lambda)$.
  We prove the CRS Indistinguishability of $\Sigma_{L_0 \vee L_1}$ through the following hybrids game, and we denote the output of the adversary in $\mathcal{H}_i$ with $out^{\mathcal{H}_i}$, to show:

$$\left| \Pr\left[out^{\mathcal{H}_0}\right] - \Pr\left[out^{\mathcal{H}_1}\right] \right| \leq \nu(\lambda)$$
$$\left| \Pr\left[out^{\mathcal{H}_1}\right] - \Pr\left[out^{\mathcal{H}_2}\right] \right| \leq \nu(\lambda)$$

  We note that the $out^{\mathcal{H}_0}$ corresponds to the output of $\mathcal{A}$ where $crs^{\mathsf{Real}}_{L_0 \vee L_1}$ is used, and the $out^{\mathcal{H}_2}$ corresponds to the output of $\mathcal{A}$ where $crs_{L_0 \vee L_1}$ generated from $\mathsf{TrapGen}_{L_0 \vee L_1}$ is used.
  $\mathcal{H}_0$:

$$\begin{aligned}
&crs_{L_0} \leftarrow\!\!\$\ \mathsf{Gen}_{L_0}(1^\lambda)\\
&crs_{L_1} \leftarrow\!\!\$\ \mathsf{Gen}_{L_1}(1^\lambda)\\
&crs_0 \leftarrow (crs_{L_0}, crs_{L_1})\\
&\text{return the output of } \mathcal{A}(crs_0)
\end{aligned}$$

  $\mathcal{H}_1$:

$$\begin{aligned}
&crs_{L_0} \leftarrow\!\!\$\ \mathsf{TrapGen}_{L_0}(1^\lambda, x_0, aux_{L_0})\\
&crs_{L_1} \leftarrow\!\!\$\ \mathsf{Gen}_{L_1}(1^\lambda)\\
&crs_1 \leftarrow (crs_{L_0}, crs_{L_1})\\
&\text{return the output of } \mathcal{A}(crs_1)
\end{aligned}$$

  $\mathcal{H}_2$:

$$\begin{aligned}
&crs_{L_0} \leftarrow\!\!\$\ \mathsf{TrapGen}_{L_0}(1^\lambda, x_0, aux_{L_0})\\
&crs_{L_1} \leftarrow\!\!\$\ \mathsf{TrapGen}_{L_1}(1^\lambda, x_1, aux_{L_1})\\
&crs_2 \leftarrow (crs_{L_0}, crs_{L_1})\\
&\text{return the output of } \mathcal{A}(crs_2)
\end{aligned}$$

- $\mathcal{H}_0 \approx \mathcal{H}_1$: If there exists a PPT adversary $\mathcal{A}$ that can distinguish between $\mathcal{H}_0$ and $\mathcal{H}_1$, we can construct an adversary $\mathcal{A}'$ that can break CRS Indistinguishability of Σ_{L_0} through the following reduction:
  * $\mathcal{A}'$ queries the challenger of the CRS Indistinguishability of Σ_{L_0} that sends back crs_{L_0}
  * $\mathcal{A}'$ samples crs_{L_1} by using Gen_{L_1}
  * $\mathcal{A}'$ sends (crs_{L_0}, crs_{L_1}) to $\mathcal{A}$
  * $\mathcal{A}'$ outputs the output of $\mathcal{A}$

  We now observe that if the challenger uses Gen_{L_0} to sample crs_{L_0}, we are in $\mathcal{H}_0$, otherwise, we are in $\mathcal{H}_1$. This implies $\mathcal{H}_0 \approx \mathcal{H}_1$.
- We can use similar reduction to show that $\mathcal{H}_1 \approx \mathcal{H}_2$.
  Now we can conclude that $\mathcal{H}_0 \approx \mathcal{H}_1 \approx \mathcal{H}_2$, so the CRS generated by $\mathsf{Gen}_{L_0 \vee L_1}$ is indistinguishable from the CRS generated by $\mathsf{TrapGen}_{L_0 \vee L_1}$.

– Correctness: Finished in Lemma 1.

The WI property instead comes immediately from the results of [CDS94] (since the WI proof only relies on the protocol being HVZK). □

2.4 Multi-theorem, adaptive non-interactive proofs

We recall that our notion of multi-theorem zero-knowledge is with respect to a single stateful prover. We now state the formal definition we consider.

Definition 5 (Adaptive Multi-Theorem Zero Knowledge). *Assuming we have a non-interactive protocol $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ for an NP language L with corresponding relation R_L. Π is adaptive multi-theorem zero knowledge if for any PPT algorithm $\mathcal{A}$, there exists a PPT simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$, running in (expected) polynomial time, such that for polynomial bounded q:*

$$\left| \Pr\left[\mathsf{Expt}_{\Pi,\mathsf{Sim},\mathcal{A}}(1^{|x|}, 1^\lambda) = 1\right] - \frac{1}{2} \right| \leq \nu(\lambda)$$

The experiment $\mathsf{Expt}_{\Pi,\mathsf{Sim},\mathcal{A}}(1^{|x|}, 1^\lambda)$ is defined as follows:

$\mathsf{Expt}_{\Pi,\mathsf{Sim},\mathcal{A}}(1^{|x|}, 1^\lambda)$:

$b \leftarrow\$ \{0,1\}, q \leftarrow 0, State_{\mathcal{A}} \leftarrow \emptyset, crs_0 \leftarrow\$ \mathsf{Setup}(1^{|x|}, 1^\lambda), (crs_1, \tau_{\mathsf{Sim}_1}) \leftarrow\$ \mathsf{Sim}_0(1^{|x|}, 1^\lambda)$

$st \leftarrow Prover(1^\lambda, crs)$

repeat

$q \leftarrow q + 1, (State_{\mathcal{A}}, x, w) \leftarrow\$ \mathcal{A}(1^\lambda, crs_b, State_{\mathcal{A}})$

if $(x, w) \in R_L$ *then* $\pi_0 \leftarrow\$ \mathsf{P}(crs_0, st, x, w), \pi_1 \leftarrow\$ \mathsf{Sim}_1(crs_1, \tau_{\mathsf{Sim}_1}, x)$

else $\pi_0 \leftarrow \pi_1 \leftarrow \emptyset$

$(State_{\mathcal{A}}, cont, d) \leftarrow\$ \mathcal{A}(1^\lambda, State_{\mathcal{A}}, \pi_b)$

until $cont = \mathsf{false}$

return $b = d$

Definition 6 (Witness Indistinguishability). *Assuming we have an interactive protocol $\Sigma_L = (\mathsf{Gen}_L, \mathsf{P}_L, \mathsf{V}_L)$ for NP language L. Σ_L is Witness Indistinguishable for relation R_L if, every malicious verifier $\mathsf{V}_L^\star$, s.t. for all x, w, w' with $(x, w) \in R_L$ and $(x, w') \in R_L$, it holds that:*

$$\left| \Pr\left[\mathsf{V}_L^\star(x, \pi_0) = 1 \mid \pi_0 \leftarrow\!\$\ \mathsf{P}_L(x, w)\right] - \Pr\left[\mathsf{V}_L^\star(x, \pi_1) = 1 \mid \pi_1 \leftarrow\!\$\ \mathsf{P}_L(x, w')\right] \right| \leq \nu(\lambda)$$

2.5 Semi-adaptive soundness

We now introduce a new notion of soundness that we call *semi-adaptive soundness*. Informally, we see every theorem x as divided into two parts (α, β), and we require the adversary to specify α before the sampling of the CRS, whereas β can be adaptively chosen from the adversary. More formally:

Definition 7 (Semi-Adaptive Soundness). *Given 2 sets $S_1 \subseteq \{0,1\}^\star, S_2 \subseteq \{0,1\}^\star$, and the NP language $L = \{(\alpha, \beta) \mid \alpha \in S_1 \wedge \beta \in S_2 \wedge \phi(\alpha, \beta) = 1\}$ defined over some predicate ϕ. Assuming we have a non-interactive protocol $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ for an NP language L with corresponding relation R_L. Π is semi-adaptive sound if for any $\alpha \in S_1$ and for any PPT prover $\mathsf{P}^\star$, it holds that:*

$$Pr_\alpha\Big[(\alpha, \beta) \notin L \wedge \mathsf{V}(crs, (\alpha, \beta), \pi) = 1 \mid \alpha \in S_1 \wedge \beta \in S_2;$$
$$crs \leftarrow\!\$\ \mathsf{Setup}(1^{|x|}, 1^\lambda); (\pi, \beta) \leftarrow \mathsf{P}^\star(crs, \alpha)\Big] \leq \nu(\lambda).$$

2.6 Semi-instance-dependant (SID) trapdoor sigma-protocol

We introduce an extension of the notion of trapdoor sigma-protocols we denote as *semi-instance-dependant trapdoor sigma-protocol*. Informally, similar to semi-adaptive soundness defined above, we divided every theorem x into 2 parts (α, β), and the $\mathsf{TrapGen}$ and $\mathsf{BadChallenge}$ algorithms of the semi-instance-dependant trapdoor sigma-protocol will take α other than the whole theorem x.

Definition 8 (Semi-instance-depandant trapdoor sigma-protocol). *Given $S_1 \subseteq \{0,1\}^\star, S_2 \subseteq \{0,1\}^\star$, and the NP language $L = \{(\alpha, \beta) \mid \alpha \in S_1 \wedge \beta \in S_2 \wedge \phi(\alpha, \beta) = 1\}$ defined over some predicate ϕ. We say that a sigma-protocol $\Sigma = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ with bad-challenge function f is a semi-instance-dependant trapdoor sigma-protocol if there are PPT algorithms $\mathsf{TrapGen}$, $\mathsf{BadChallenge}$ with the following syntax.*

- $\mathsf{TrapGen}(1^\lambda, \alpha, aux)$ *takes as input the unary representation of the security parameter, the first part of the instance α, and an auxiliary input aux. It outputs a common reference string crs along with a trapdoor τ.*
- $\mathsf{BadChallenge}(\tau, crs, \alpha, a)$ *takes as input a trapdoor τ, common reference string crs, the first part of the instance α, and first message a. It outputs a challenge c.*

We additionally require the following properties:

- ***CRS Indistinguishability**: For any (α, aux), an honestly generated common reference string crs is computationally indistinguishable from a common reference string output by $\mathsf{TrapGen}(1^\lambda, \alpha, aux)$.*
- ***Correctness**: For every instance $x \notin L$, there exists an auxiliary input aux such that for all $(crs, \tau) \leftarrow\!\!\$\ \mathsf{TrapGen}(1^\lambda, \alpha, aux)$, we have that $\mathsf{BadChallenge}(\tau, crs, \alpha, a) = f(crs, x, a)$.*

We argue that the OR composition of [CDS94] applied on a SID trapdoor sigma-protocol and an ID trapdoor sigma-protocol yields a new SID for the OR relation. More formally, assuming the existence of an ID trapdoor sigma-protocol $\Sigma_{L_0} = (\mathsf{Gen}_{L_0}, \mathsf{P}_{L_0}, \mathsf{V}_{L_0})$ for NP language L_0 and a SID trapdoor sigma-protocol $\Sigma_{L_1} = (\mathsf{Gen}_{L_1}, \mathsf{P}_{L_1}, \mathsf{V}_{L_1})$ for NP language $L_1 = \{(\alpha, \beta) \mid \alpha \in S_1 \wedge \beta \in S_2 \wedge \phi(\alpha, \beta) = 1\}$, then the application of the compiler of [CDS94] on Σ_{L_0} and Σ_{L_1} will yield a SID trapdoor sigma-protocol $\Sigma_{L_0 \vee L_1}$, such that the following lemma holds.

Lemma 3. *$\Sigma_{L_0 \vee L_1}$ is a semi-instance-dependant trapdoor sigma-protocol, and it is witness indistinguishable, for NP language $L = \{((\alpha, x), \beta) \mid (\alpha, x) \in S_1' \wedge \beta \in S_2' \wedge (\phi(\alpha, \beta) = 1 \vee x \in L_0)\}$, where $S_1' = S_1 \times \{0,1\}^\star$ and $S_2' = S_2$.*

Proof. The proof is nearly identical to the proof for Lemma 2.

2.7 Correlation-intractable hash functions and FS transform

Here we recall the related definitions of Correlation-Intractable Hash Family (CIHF) from [CCH+19].

Definition 9 (Hash family). *For a pair of efficiently computable functions $(n(\cdot), m(\cdot))$, a hash family with input length n and output length m is a collection $\mathcal{H} = \{h_k : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}\}_{\lambda \in \mathbb{N}, k \in \{0,1\}^{s(\lambda)}}$ of keyed hash functions, along with a pair of PPT algorithms specified as follows: (i) $\mathcal{H}.\mathsf{Gen}(1^\lambda)$ outputs a hash key $k \in \{0,1\}^{s(\lambda)}$; (ii) $\mathcal{H}.\mathsf{H}(k, x)$ computes the function $h_k(x)$.*

Definition 10 (Correlation intractability). *For a given relation ensemble $R := \{R_\lambda \subseteq \{0,1\}^{n(\lambda)} \times \{0,1\}^{m(\lambda)}\}$, a hash family $\mathcal{H} = \{h_k : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}\}_{\lambda \in \mathbb{N}, k \in \{0,1\}^{s(\lambda)}}$ is said to be R-correlation intractable with security (σ, δ) if for every σ-size attacker $\mathcal{A} := \{\mathcal{A}_\lambda\}$:*

$$\Pr\left[(x, h_k(x)) \in R_\lambda : \ k \leftarrow\!\!\$\ \mathcal{H}.\mathsf{Gen}(1^\lambda); x \leftarrow\!\!\$\ \mathcal{A}(k)\right] = O(\delta(\lambda)).$$

We say that $\mathcal{H}$ is R-correlation intractable if it is R-correlation intractable with security $(\lambda^c, \lambda^{-c})$ for all constants $c > 1$.

Definition 11 (Sparsity). *For any relation ensemble $R := \{R_\lambda \subseteq \{0,1\}^{n(\lambda)} \times \{0,1\}^{m(\lambda)}\}_\lambda$, we say that R is $\rho(\cdot)$-sparse if for all $\lambda \in \mathbb{N}$ and for any $x \in \{0,1\}^{n(\lambda)}$ it holds that $(x, y) \in R_\lambda$ with probability at most $\rho(\lambda)$ over the choice of $y \leftarrow\!\!\$\ \{0,1\}^{m(\lambda)}$. When ρ is a negligible function, we say that R is sparse.*

Efficiently Searchable Relations. In this work, we will need hash families to achieve correlation intractability for relations R with a unique output $y = f(x)$ associated to each input x, and such that $y = f(x)$ is an efficiently computable function of x.

Definition 12 (Unique output relation). *We say that a relation R is a unique output relation if for every input x, there exists at most one output y such that $(x, y) \in R$.*

Definition 13 (Efficiently searchable relation). *We say that a (necessarily unique-output) relation ensemble R is searchable in (non-uniform) time t if there exists a function $f = f_R : \{0,1\}^* \to \{0,1\}^*$ computable in (non-uniform) time t such that for any input x, if $(x, y) \in R$ then $y = f(x)$; that is, $f(x)$ is the unique y such that $(x, y) \in R$, provided that such a y exists. We say that R is efficiently searchable if it is searchable in time $\mathsf{poly}(n)$.*

Programmability. The following property turns out to be very useful to prove the zero-knowledge property of non-interactive proofs derived using correlation intractable hash families.

Definition 14 (1-universality). *We say that a hash family $\mathcal{H}$ is 1-universal if for any $\lambda \in \mathbb{N}$, input $x \in \{0,1\}^{n(\lambda)}$, and output $y \in \{0,1\}^{m(\lambda)}$, we have $\Pr\left[h_k(x) = y : k \leftarrow\!\!\$\ \mathcal{H}.\mathsf{Gen}(1^\lambda)\right] = 2^{-m(\lambda)}$.*

We say that a hash family $\mathcal{H}$ is programmable if it is 1-universal, and if there exists an efficient sampling algorithm $\mathsf{Samp}(1^\lambda, x, y)$ that samples from the conditional distribution $k \leftarrow\!\!\$\ \mathcal{H}.\mathsf{Gen}(1^\lambda) | h_k(x) = y$.

We recall the theorem from [CCH+19] that we use in our work:

Theorem 1 ([CCH+19]). *Suppose that $\mathcal{H}$ is a hash family that is correlation-intractable for all subexponentially sparse relations that are searchable in time T. Moreover, suppose that $\Sigma = (\mathsf{Gen}, \mathsf{P}, \mathsf{V}, \mathsf{TrapGen}, \mathsf{BadChallenge})$ is an instance-dependent trapdoor sigma-protocol with $2^{-\lambda^\epsilon}$ soundness for some $\epsilon > 0$, such that $\mathsf{BadChallenge}(\tau, crs, x, a)$ is computable in time T. Then, $\mathcal{H}$ soundly instantiates the Fiat-Shamir heuristic for Σ.*

A note on NIZK from ID trapdoor sigma-protocol. Assuming the existence of an ID trapdoor sigma-protocol Σ_L for NP language L, then the application of Theorem 1 on Σ_L will yield a sound NIZK protocol Π_L.

In our work, we also make use of the following lemmas. The application of Theorem 1 on $\Sigma_{L_0 \vee L_1}$ (from Lemma 2) will yield a NIZK protocol $\Pi_{L_0 \vee L_1}$, such that the following lemma holds.

Lemma 4. *$\Pi_{L_0 \vee L_1}$ is sound and WI.*

Proof. By Lemma 2, we know $\Sigma_{L_0 \vee L_1}$ is an ID trapdoor sigma-protocol, and by applying Theorem 1, we know $\Pi_{L_0 \vee L_1}$ is sound.

Also, by Lemma 2, we know $\Sigma_{L_0 \vee L_1}$ is WI. By Claim 1 in [YZ06], we know that a non-interactive Σ_{OR} protocol from the OR-composition protocol is WI if the Random Oracle model is replaced by any real hash functions.

Because $\mathcal{H}$ is a hash family, which is also a real hash function, $\Pi_{L_0 \vee L_1}$ is witness indistinguishable (WI). The proof is nearly identical to the proof in Theorem 5 of [CPSV16]. □

For Lemma 5, it states that FS transform with CIHF applied on any SID trapdoor sigma-protocols will yield a semi-adaptive sound NIZK.

Lemma 5. *Let Σ_L be a semi-instance-dependant trapdoor sigma-protocol, for language $L = \{(\alpha, \beta) \mid \alpha \in S_\alpha \wedge \beta \in S_\beta \wedge \phi(\alpha, \beta) = 1\}$. Then, NIZK Π_L obtained by applying FS transform with a CIHF $\mathcal{H}$ on Σ_L, is semi-adaptive sound, for language L.*

Proof. This proof is similar to Canetti et al. 's proofs for Theorem 1. Assuming Π_L is not semi-adaptive sound. It means there exists a PPT algorithm $\mathcal{A}$, s.t.:

$$Pr_\alpha \Big[(\alpha, \beta) \notin L \wedge \mathsf{V}_L(crs, (\alpha, \beta), \pi) = 1 \mid \alpha \in S_1 \wedge \beta \in S_2; \\ crs \leftarrow\!\!\$\ \mathsf{Setup}(1^{|(\alpha,\beta)|}, 1^\lambda); (\pi, \beta) \leftarrow\!\!\$\ \mathsf{P}^\star(crs, \alpha) \Big] \geq \delta(\lambda).$$

Then we can construct an adversary $\mathcal{A}_{CI}$ to break CI of $\mathcal{H}$ for relation $R_{\tau,crs,\alpha}$, where $\alpha \in S_\alpha$, The relation $R_{\tau,crs,\alpha}$ is as follows:

$$R_{\tau,crs,\alpha} = \{(a, c) : c = \mathsf{BadChallenge}_L(\tau, crs, \alpha, a)\}$$

Where $(crs, \tau) \leftarrow\!\!\$\ \mathsf{TrapGen}_L(1^\lambda, \alpha, aux)$.

The adversary $\mathcal{A}_{CI}$ is as follows:

$\mathcal{A}_{CI}(k, crs, \alpha)$:
  $(\pi, \beta) \leftarrow \mathcal{A}((crs, k), \alpha)$
  parsing π as (a, z)
  return a

Now we have the following observation:

– $\mathcal{A}$ works correctly. We observe that the input crs to $\mathcal{A}_{CI}$ is from $\mathsf{TrapGen}$, but $\mathcal{A}$ requires the input crs from Gen. If $\mathcal{A}$'s behavior is different, we can use it to break the CRS Indistinguishability of Σ_L, and we will demonstrate it through the following hybrid game. We denote the output of $\mathcal{A}$ in $\mathcal{H}_i$ with $out^{\mathcal{H}_i}$, and we want to prove:

$$\Big| \Pr\big[out^{\mathcal{H}_0}\big] - \Pr\big[out^{\mathcal{H}_1}\big] \Big| \leq \nu$$

We note that the $out^{\mathcal{H}_0}$ corresponds to the output of $\mathcal{A}$ where crs generated by Gen is used, and the $out^{\mathcal{H}_1}$ corresponds to the output of $\mathcal{A}$ where crs generated by $\mathsf{TrapGen}$ is used. Then the hybrids are:

- $\mathcal{H}_0$:

$$crs \leftarrow\!\$\ \mathsf{Gen}(1^\lambda)$$
$$k \leftarrow\!\$\ \mathcal{H}.\mathsf{Gen}(1^\lambda)$$
$$\text{return the output of } \mathcal{A}((crs, k), \alpha)$$

- $\mathcal{H}_1$:

$$(crs, \tau) \leftarrow\!\$\ \mathsf{TrapGen}(1^\lambda, \alpha, aux)$$
$$k \leftarrow\!\$\ \mathcal{H}.\mathsf{Gen}(1^\lambda)$$
$$\text{return the output of } \mathcal{A}((crs, k), \alpha)$$

If $\mathcal{A}$'s behaviors are different, then we can construct an adversary $\mathcal{A}_{crs}$ to break the CRS Indistinguishability of Σ_L through the following reduction:

- $\mathcal{A}_{crs}$ queries the challenger of the CRS Indistinguishability of Σ_L, that sends back crs_L
- $\mathcal{A}_{crs}$ samples the hash key k, and α
- $\mathcal{A}_{crs}$ sends $((crs, k), \alpha)$ to $\mathcal{A}$, and outputs the output of $\mathcal{A}$

We now observe that if the challenger uses Gen, we are in $\mathcal{H}_0$, otherwise, we are in $\mathcal{H}_1$. It implies $\mathcal{H}_0 \approx \mathcal{H}_1$. Therefore, $\mathcal{A}$ works correctly.

– Output of $\mathcal{A}$ make V_L accept with non-negligible probability, and it means that we find a valid a when $(\alpha, \beta) \notin L$. Because Σ_L has $2^{-\lambda^\epsilon}$ soundness, $R_{\tau,crs,\alpha}$ is subexponential sparse. Besides, $\mathsf{BadChallenge}_L$ is an efficient algorithm, by Definition 4. Therefore, it contradicts the assumption of $\mathcal{H}$

□

The existence of the SID trapdoor sigma-protocols. In [CPV20] the authors observe that it is possible to extract the unique bad-challenge for well-known Chaum-Pedersen sigma-protocols [CP93] for DH tuples that we denote with Σ_{DH} (we recall it in Figure 2, where $crs = \emptyset$).

In particular, the authors show how to extract the bad-challenge of the 1-bit challenge version of the sigma-protocol Σ_{DH} for DH tuples. We show that the parallel repetition version Σ^t_{DH} is a SID trapdoor sigma-protocol for $L_{\mathsf{DH}} = \{(g, h, X, Y) \mid (g, h) \in S_1 \wedge (X, Y) \in S_2 \wedge \phi(g, h, X, Y) = 1\}$, where $S_1 = \{(g, g^x) \in G \times G \mid x \in \mathbb{Z}_p\}$, $S_2 = \{(h, h^y) \in G \times G \mid y \in \mathbb{Z}_p\}$, and $\phi(g, h, X, Y) = 1$ if and only if $\exists w \in \mathbb{Z}_p : X = g^w \wedge Y = h^w$. Formally:

Theorem 2. *Let Σ^t_{DH} be the parallel repetition version of Σ_{DH}, with the number of repetition t. Then, Σ^t_{DH} is a semi-instance-dependant trapdoor sigma-protocol, for L_{DH}.*

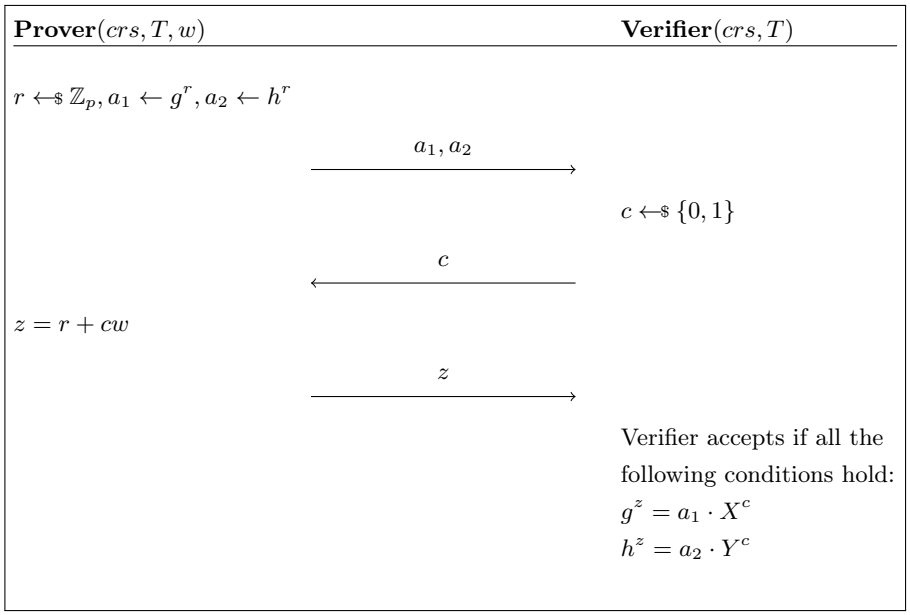

Fig. 2. Sigma-protocol Σ_{DH} for L_{DH}

Proof. Because we know Σ_{DH} is a sigma-protocol for $L_{\mathsf{DH}} = \{(g,h,X,Y) \mid (g,h) \in S_1 \wedge (X,Y) \in S_2 \wedge \phi(g,h,X,Y) = 1\}$, where $S_1 = \{(g,g^x) \in G \times G \mid x \in \mathbb{Z}_p\}$, $S_2 = \{(h,h^y) \in G \times G \mid y \in \mathbb{Z}_p\}$, and $\phi(g,h,X,Y) = 1$ if and only if $\exists w \in \mathbb{Z}_p : X = g^w \wedge Y = h^w$, by applying Lemma 1 in [Dam10], we have the following claim:

Claim. Σ^t_{DH} is a sigma-protocol.

By the claim above, we know Σ^t_{DH} is a sigma-protocol, so we can only prove the properties for the SID trapdoor sigma-protocol.

The $\mathsf{TrapGen}_{\mathsf{DH}}$ algorithm takes the following inputs:

- 1^λ: The unary representation of the security parameter.
- α: (g,h) from the tuple $T = (g,h,X,Y)$
- aux: x from $g^x = h$

and $\mathsf{TrapGen}_{\mathsf{DH}}$ outputs $crs = \emptyset$ and $\tau = aux$.

Before we describe the construction of $\mathsf{BadChallenge}_{\mathsf{DH}}$, here we describe how to extract the unique 1-bit bad-challenge from Σ_{DH}, by using PPT $\mathsf{E}_{\mathsf{uni}}(\tau, a_1, a_2)$ algorithm, where τ is the trapdoor and a_1, a_2 are the first round message of Σ_{DH}.

It works as follows:

$$
\begin{aligned}
&\mathsf{E}_{\mathsf{uni}}(\tau, a_1, a_2) : \\
&\quad \text{If } a_1^{\tau} = a_2 \text{ return } 0 \\
&\quad \text{If } a_1^{\tau} \neq a_2 \text{ return } 1
\end{aligned}
$$

Then we denote the transcript for i-th repetition with (a_1^i, a_2^i, c_i, z_i). By having the output of $\mathsf{TrapGen}_{\mathsf{DH}}$, and the $\mathsf{E}_{\mathsf{uni}}$ algorithm, we have the following $\mathsf{BadChallenge}_{\mathsf{DH}}$ algorithm:

$$
\begin{aligned}
&\mathsf{BadChallenge}(\tau, crs, \alpha, a) : \\
&\quad c_1 = \mathsf{E}_{\mathsf{uni}}(\tau, a_1^1, a_2^1) \\
&\quad ... \\
&\quad c_t = \mathsf{E}_{\mathsf{uni}}(\tau, a_1^t, a_2^t) \\
&\quad c = (c_1 || c_2 || ... || c_t) \\
&\quad \text{return } c
\end{aligned}
$$

where $a = ((a_1^1, a_2^1), ..., (a_1^t, a_2^t))$ is the first round message of Σ_{DH}^t, and x is the tuple T.

By the construction of $\mathsf{E}_{\mathsf{uni}}$ and $\mathsf{BadChallenge}_{\mathsf{DH}}$, we know $\mathsf{BadChallenge}_{\mathsf{DH}}$ is a PPT algorithm. Then we prove the CRS Indistinguishability and Correctness:

- CRS Indistinguishability:

  Because CRS for Σ_{DH}^t is an empty string, the honestly generated CRS is computationally indistinguishable from CRS computed by $\mathsf{TrapGen}_{\mathsf{DH}}$

- Correctness: Completeness of $\mathsf{E}_{\mathsf{uni}}$ is already proven in paper [CPV20]. We show details with mathematical calculations here.

  Assuming we have a non-DH tuple $T = (g, h, X = g^{w_1}, Y = h^{w_2})$, where $g^x = h$ and $w_1 \neq w_2$. Then we need to prove for any choice of first round message a of Σ_{DH}^t, there is at most 1 challenge c, to make transcript (a, c, z) be accepted.

  Considering c_i (the i-th bit of challenge), and we have $\tau = x$:

- If $(a_1^i)^\tau = a_2^i$, proving that no valid third round message z for $c_i = 1$:

$$\begin{cases} g^z & = a_1^i X \\ h^z & = a_2^i Y \end{cases}$$
$$\rightarrow \begin{cases} g^{(z-w_1)} & = a_1^i \\ h^{(z-w_2)} & = a_2^i \end{cases}$$
$$\rightarrow \begin{cases} g^{(z-w_1)} & = a_1^i \\ g^{x(z-w_2)} & = (a_1^i)^x \end{cases}$$
$$\rightarrow \begin{cases} g^{(z-w_1)} & = a_1^i \\ g^{x(z-w_2)} & = g^{x(z-w_1)} \end{cases}$$
$$\rightarrow \begin{cases} g^{(z-w_1)} & = a_1^i \\ x(w_1 - w_2) & = 0 \end{cases}$$

  * We know $x \neq 0$ and $w_1 \neq w_2$
  * It is impossible to have a valid z when $c_i = 1$. Therefore, no accepting transcripts.
- When $(a_1^i)^\tau = a_2^i$, and $c_i = 0$:

$$\begin{cases} g^z & = a_1^i \\ h^z & = a_2^i \end{cases}$$
$$\rightarrow \begin{cases} g^z & = a_1^i \\ g^{xz} & = (a_1^i)^x \end{cases}$$
$$\rightarrow g^z = a_1^i$$

  * Because a is fixed, so a_1^i is fixed, and there is at most 1 z to make the equation hold.
- Then when $(a_1^i)^\tau \neq a_2^i$, if challenge $c = 1$, we have:

$$\begin{cases} g^z & = a_1^i X \\ h^z & = a_2^i Y \end{cases}$$
$$\rightarrow \begin{cases} g^z & = a_1^i X \\ g^{xz} & = a_2^i Y \end{cases}$$
$$\rightarrow \begin{cases} g^z & = a_1^i X \\ g^{xz} & = a_2^i Y \end{cases}$$
$$\rightarrow \begin{cases} g^z & = a_1^i X \\ (a_1^i X)^x & = a_2^i Y \end{cases}$$
$$\rightarrow \begin{cases} g^z & = a_1^i X \\ (a_1^i)^x & = a_2^i h^{w_2 - w_1} \end{cases}$$

* For given (a_1^i, a_2^i), this equation is possible to hold, which means an accepting transcript may exist.

- Then when $(a_1^i)^\tau \neq a_2^i$, if challenge $c = 0$, we have:

$$\begin{cases} g^z &= a_1^i \\ h^z &= a_2^i \end{cases}$$
$$\rightarrow \begin{cases} g^z &= a_1^i \\ (a_1^i)^x &= a_2^i \end{cases}$$

  * Because we know $(a_1^i)^x \neq a_2^i$
  * There is no accepting transcript.

By above illustrations, we know the algorithm $\mathsf{E}_{\mathsf{uni}}(\tau, a_1^i, a_2^i)$ can output c_i, which is the unique bad-challenge for (a_1^i, a_2^i). The $\mathsf{E}_{\mathsf{uni}}$ algorithm is complete. Then, assuming the Correctness does not hold, which means the $\mathsf{BadChallenge}_{\mathsf{DH}}$ does not output the unique bad-challenge. By the construction of Σ_{DH}^t, we know that, if the output is not the unique bad-challenge, then at least the transcript for one of the repetitions is not accepted. Formally, we denote the transcripts that are not accepted as $\pi_i = ((a_1^i, a_2^i), c_i, z_i)$. However, if π_i is not accepted, it means $\mathsf{E}_{\mathsf{uni}}$ does not find the unique bad-challenge for (a_1^i, a_2^i), which contradicts the completeness of $\mathsf{E}_{\mathsf{uni}}$.

□

One of the main tools we rely on is a SID trapdoor sigma-protocol for the language of the non-DH tuple. In particular, we need to construct a protocol Σ_{NDH} for the language $L_{\mathsf{NDH}} = \{(g, h, X, Y) \mid (g, h) \in S_1 \wedge (X, Y) \in S_2 \wedge \phi(g, h, X, Y) = 1\}$, where $S_1 = \{(g, g^x) \in G \times G \mid x \in \mathbb{Z}_p\}$, $S_2 = \{(h, h^y) \in G \times G \mid y \in \mathbb{Z}_p\}$, and $\phi(g, h, X, Y) = 1$ if and only if $\exists w, w' \in \mathbb{Z}_p : X = g^w \wedge Y = h^{w'} \wedge w \neq w'$. At a high level, our protocol works as follows. The prover computes a commitment of a random value $b \in \{0,1\}^\tau$. The commitment is equivocal when $T \in L_{\mathsf{NDH}}$ and it is binding (and extractable) otherwise. The prover sends the commitment of b to the verifier, who replies with a uniformly random $c \in \{0,1\}$. In the third round, the prover will equivocate the commitment to an opening of c, and send the opening information to the verifier. We recall that the honest prover can always equivocate the commitment since $T \in L_{\mathsf{NDH}}$.

This protocol is sound since when $T \notin L_{\mathsf{NDH}}$, the probability of the prover providing a valid opening for c is $2^{-\tau}$. To extract the bad-challenge, we will rely on the fact that the commitment is extractable when $T \notin L_{\mathsf{NDH}}$. In particular, we prove that it is possible to extract the bad-challenge for a proof computed with respect to a tuple $T = (g, h, X, Y)$, having access only to the discrete logarithm of h. This is the reason why our protocol is only semi-adaptive and not fully adaptive (i.e., if the entire tuple was chosen by the adversary then the extractor would have no access to the discrete logarithm of h).

One nice feature of the protocol we have described is that for a challenge of size $\tau = \log \lambda$, where λ is the security parameter, prover and verifier need to

perform only 4 exponentiations each, and we give the efficiency analysis later. We see Σ_{NDH} as a result of independent interest. Previous to our work, it was already known how to construct a trapdoor sigma protocol with similar performance, but ours is the first protocol to have such performance while being a SID trapdoor sigma-protocol. In particular, we note that in [LNPY22], the authors give a construction of trapdoor sigma-protocol for the language of DH (hence, also for the language of non-DH) tuples with similar performance as ours. Unfortunately, it is not clear how to prove that the protocol proposed in [LNPY22] is also a SID trapdoor sigma-protocols.

We propose the formal description of our protocol $\Sigma_{\mathsf{NDH}} = (\mathsf{Gen}_{\mathsf{NDH}}, \mathsf{P}_{\mathsf{NDH}}, \mathsf{V}_{\mathsf{NDH}})$ in Figure 3, where the $crs = \emptyset$.

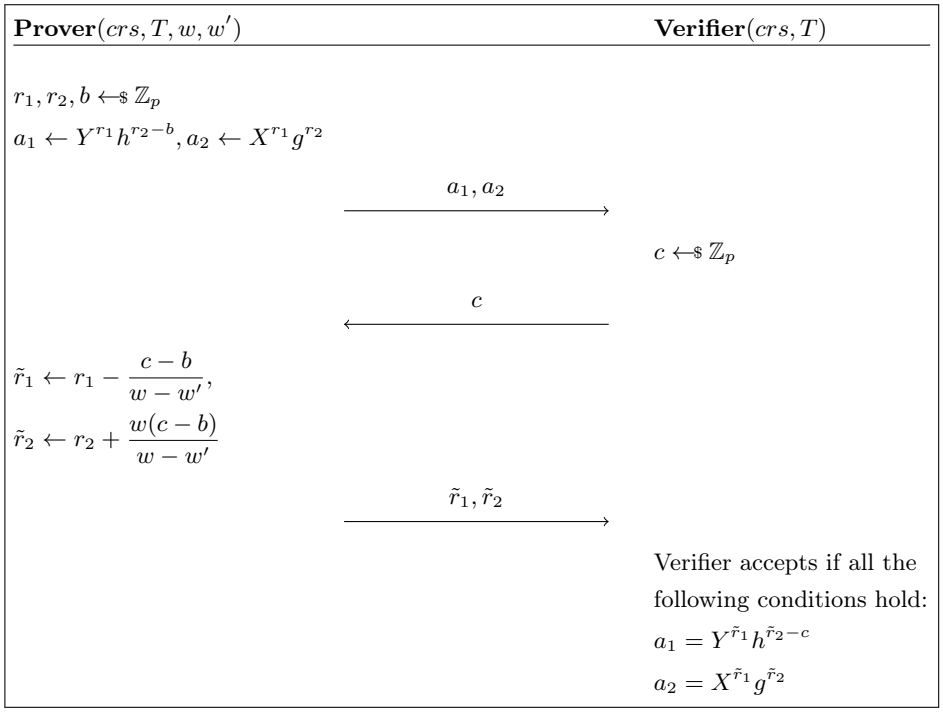

Fig. 3. The protocol for L_{NDH}

Lemma 6. *Σ_{NDH} is a sigma-protocol for language L_{NDH}.*

Proof. **Completeness**:

– For new $\tilde{r}_1, \tilde{r}_2$:

$$
\begin{aligned}
&\begin{cases} Y^{\tilde{r}_1} h^{\tilde{r}_2 - c} &= Y^{r_1} h^{r_2 - b} \\ X^{\tilde{r}_1} g^{\tilde{r}_2} &= X^{r_1} g^{r_2} \end{cases} \\
\rightarrow &\begin{cases} h^{w'\tilde{r}_1 + \tilde{r}_2 - c} &= h^{w' r_1 + r_2 - b} \\ g^{w\tilde{r}_1 + \tilde{r}_2} &= g^{w r_1 + r_2} \end{cases} \\
\rightarrow &\begin{cases} w'\tilde{r}_1 + \tilde{r}_2 - c &= w' r_1 + r_2 - b \\ w\tilde{r}_1 + \tilde{r}_2 &= w r_1 + r_2 \end{cases} \\
\rightarrow &\begin{cases} \tilde{r}_1 &= r_1 + \frac{r_2 - \tilde{r}_2}{w} \\ w'(r_1 + \frac{r_2 - \tilde{r}_2}{w}) + \tilde{r}_2 &= w' r_1 + r_2 + c - b \end{cases}
\end{aligned}
$$

Then we have:

$$
\begin{aligned}
w'(r_1 + \frac{r_2 - \tilde{r}_2}{w}) + \tilde{r}_2 &= w' r_1 + r_2 + c - b \\
w' r_1 + \frac{w'}{w} r_2 - \frac{w'}{w}\tilde{r}_2 + \tilde{r}_2 &= w' r_1 + r_2 + c - b \\
(1 - \frac{w'}{w})\tilde{r}_2 &= (1 - \frac{w'}{w}) r_2 + c - b \\
\tilde{r}_2 &= r_2 + \frac{w(c-b)}{w - w'}
\end{aligned}
$$

Therefore:

$$
\begin{aligned}
&\begin{cases} \tilde{r}_1 &= r_1 + \frac{r_2 - \tilde{r}_2}{w} \\ \tilde{r}_2 &= r_2 + \frac{w(c-b)}{w-w'} \end{cases} \\
\rightarrow &\begin{cases} \tilde{r}_1 &= r_1 + \frac{r_2 - (r_2 + \frac{w(c-b)}{w-w'})}{w} \\ \tilde{r}_2 &= r_2 + \frac{w(c-b)}{w-w'} \end{cases} \\
\rightarrow &\begin{cases} \tilde{r}_1 &= r_1 - \frac{c-b}{w-w'} \\ \tilde{r}_2 &= r_2 + \frac{w(c-b)}{w-w'} \end{cases}
\end{aligned}
$$

Optimal soundness: Assume by contradiction that we have 2 accepting transcripts $\tau_\alpha = ((a_1, a_2), c_\alpha, (\tilde{r}_1^\alpha, \tilde{r}_2^\alpha))$, and $\tau_\beta = ((a_1, a_2), c_\beta, (\tilde{r}_1^\beta, \tilde{r}_2^\beta))$, where $c_\alpha \neq c_\beta$, and the tuple $T \notin L_{\mathsf{NDH}}$. We do not know the relationship between

$(\tilde{r}_1^\alpha, \tilde{r}_2^\alpha)$ and $(\tilde{r}_1^\beta, \tilde{r}_2^\beta)$ Then we have the following equations:

$$\begin{cases} Y^{\tilde{r}_1^\alpha} h^{\tilde{r}_2^\alpha - c_\alpha} & = a_1 = Y^{\tilde{r}_1^\beta} h^{\tilde{r}_2^\beta - c_\beta} \\ X^{\tilde{r}_1^\alpha} g^{\tilde{r}_2^\alpha} & = a_2 = X^{\tilde{r}_1^\beta} g^{\tilde{r}_2^\beta} \end{cases}$$

$$\rightarrow \begin{cases} \frac{(X^{\tilde{r}_1^\alpha} g^{\tilde{r}_2^\alpha})^x}{h^{c_\alpha}} & = a_1 = \frac{(X^{\tilde{r}_1^\beta} g^{\tilde{r}_2^\beta})^x}{h^{c_\beta}} \\ X^{\tilde{r}_1^\alpha} g^{\tilde{r}_2^\alpha} & = a_2 = X^{\tilde{r}_1^\beta} g^{\tilde{r}_2^\beta} \end{cases}$$

$$\rightarrow \begin{cases} \frac{a_2^x}{h^{c_\alpha}} & = a_1 = \frac{a_2^x}{h^{c_\beta}} \\ X^{\tilde{r}_1^\alpha} g^{\tilde{r}_2^\alpha} & = a_2 = X^{\tilde{r}_1^\beta} g^{\tilde{r}_2^\beta} \end{cases}$$

By the above equations, we know $a_2^x = a_2^x$, so $c_\alpha = c_\beta = f(crs, T, (a_1.a_2))$

Special HVZK:

- The simulator works as follows:

$$\begin{aligned} &\mathsf{Sim}(T, c): \\ &\quad \tilde{r}_1, \tilde{r}_2 \leftarrow\!\$\ \mathbb{Z}_p \\ &\quad a_1 \leftarrow Y^{\tilde{r}_1} h^{\tilde{r}_2 - c}, a_2 \leftarrow X^{\tilde{r}_1} g^{\tilde{r}_2} \\ &\quad a_{\mathsf{Sim}} \leftarrow (a_1, a_2), z_{\mathsf{Sim}} \leftarrow (\tilde{r}_1, \tilde{r}_2) \\ &\quad \text{return } (a_{\mathsf{Sim}}, z_{\mathsf{Sim}}) \end{aligned}$$

- Let $\tau_{\mathsf{Real}} = (a_{\mathsf{Real}}, c_{\mathsf{Real}}, z_{\mathsf{Real}})$ be the real execution transcript. We note that $(a_{\mathsf{Real}}, z_{\mathsf{Real}})$ is indistinguishable from $(a_{\mathsf{Sim}}, z_{\mathsf{Sim}})$, because:
  - In a_{Real}, (b, r_1, r_2) are uniform randomly sampled, and in a_{Sim}, $(c, \tilde{r}_1, \tilde{r}_2)$ are uniform randomly sampled, so Y^{r_1} is indistinguishable from $Y^{\tilde{r}_1}$, and $h^{r_2 - b}$ is indistinguishable from $h^{\tilde{r}_2 - c}$. Then a_1 in a_{Real} is indistinguishable from a_1 in a_{Sim}.
  - Similar proofs can be done for a_2, so a_{Real} is indistinguishable from a_{Sim}
  - for $\tilde{r}_1$ in z_{Real}, because r_1, c, b are uniform randomly sampled, $c - b$ is uniformly random. $(w - w')$ is constant for every execution, so $\frac{c-b}{w-w'}$ is uniformly random. Therefore $\tilde{r}_1$ is uniformly random, and it is indistinguishable from $\tilde{r}_1$ from a_{Sim}.
  - Similar proofs can be done for $\tilde{r}_2$, so z_{Real} is indistinguishable from z_{Sim}

□

Lemma 7. *Σ_{NDH} has a PPT extractor $\mathsf{Ext}_{\mathsf{uni}}(\alpha, \tau, a)$, where α is (g, h) from the tuple $T = (g, h, X, Y)$, τ is the trapdoor, a is the first round message, s.t. $\forall T \notin L_{\mathsf{NDH}}$, if the unique bad-challenge is c, $\mathsf{Ext}_{\mathsf{uni}}$ can extract h^c (which is also unique).*

Proof. $\mathsf{Ext}_{\mathsf{uni}}$, on input $\alpha = (g, h)$, $\tau = x$, such that $g^x = h$, $a = (a_1, a_2)$ (the first round of the sigma-protocol Σ_{NDH}), returns $h^c \leftarrow \frac{a_2^\tau}{a_1}$, where c is the bad-challenge.

$\mathsf{Ext}_{\mathsf{uni}}$ outputs the correct results due to the following observation. If we have the first round message $a = (a_1, a_2)$, and $T \notin L_{\mathsf{NDH}}$, due to the optimal soundness property, we know that there is at most one challenge c that makes the transcript (a, c, z) accepting. Then because the transcript is accepting it must be that $a_1 = Y^{\tilde{r}_1} h^{\tilde{r}_2 - c}$ and $a_2 = X^{\tilde{r}_1} g^{\tilde{r}_2}$. When $T \notin L_{\mathsf{NDH}}$, $\frac{a_2^\tau}{a_1} = \frac{X^{x\tilde{r}_1} g^{x\tilde{r}_2}}{Y^{\tilde{r}_1} h^{\tilde{r}_2}} h^c = h^c$. Because g is the generator in the cyclic group G, g^c and c are 1 to 1 mapping, $h^c = g^{xc}$ and x is fixed for every execution. It means h^c is also unique.

Claim. If the number of all the possible challenges c is bounded to $\mathsf{poly}(\lambda)$, then by using brute force, computing c from h^c is efficient (polynomial time in λ).

Lemma 8. *If the challenge c of the protocol Σ_{NDH} satisfies that $c \in \{0,1\}^{\mathcal{K}\log_2(\lambda^\epsilon)}$ for $\epsilon > 0$ and for integer $\mathcal{K} \geq 1$, then for $t = \Omega(\frac{\lambda^\epsilon}{\mathcal{K}\log_2(\lambda^\epsilon)})$, the parallel repetition version Σ^t_{NDH} is a semi-instance-dependant trapdoor sigma-protocol, for $L_{\mathsf{NDH}} = \{(g,h,X,Y) \mid (g,h) \in S_1 \wedge (X,Y) \in S_2 \wedge \phi(g,h,X,Y) = 1\}$, where $S_1 = \{(g, g^x) \in G \times G \mid x \in \mathbb{Z}_p\}$, $S_2 = \{(h, h^y) \in G \times G \mid y \in \mathbb{Z}_p\}$, and $\phi(g,h,X,Y) = 1$ if and only if $\exists w, w' \in \mathbb{Z}_p : X = g^w \wedge Y = h^{w'} \wedge w \neq w'$.*

Proof. By applying Lemma 1 in [Dam10], we know Σ^t_{NDH} is a sigma-protocol, and we only focus on proving the property for the SID trapdoor sigma-protocol.

The corresponding $\mathsf{TrapGen}_{\mathsf{NDH}}$ algorithm has the following inputs:

- 1^λ: The unary representation of the security parameter
- α: (g,h) from the tuple (g,h,X,Y)
- aux: x from $g^x = h$

Then the outputs of $\mathsf{TrapGen}_{\mathsf{NDH}}$ is $crs = \emptyset$ and $\tau = aux$.

We denote the transcript of i-th repetition as (a_1^i, a_2^i, c_i, z_i). The construction of the bad-challenge extractor $\mathsf{BadChallenge}_{\mathsf{NDH}}(\tau, crs, \alpha, a)$ is:

$\mathsf{BadChallenge}_{\mathsf{NDH}}(\tau, crs, \alpha, a)$:
  $h^{c_1} \leftarrow \mathsf{Ext}_{\mathsf{uni}}(\alpha, \tau, (a_1^1, a_2^1))$
  Brute force search on h^{c_1} to get c_1
  ...
  $h^{c_t} \leftarrow \mathsf{Ext}_{\mathsf{uni}}(\alpha, \tau, (a_1^t, a_2^t))$
  Brute force search on h^{c_t} to get c_t
  $c \leftarrow (c_1 || c_2 || ... || c_t)$
  return c

where $a = ((a_1^1, a_2^1), ..., (a_1^t, a_2^t))$ is the first round message of Σ^t_{NDH}. By Lemma 7 and the claim that brute force is efficient for small search space, we know $\mathsf{BadChallenge}_{\mathsf{NDH}}$ is a PPT algorithm.

Then we prove the CRS Indistinguishability and Correctness:

- CRS Indistinguishability:
  Because the Σ^t_{NDH}'s CRS is an empty set, the honestly generated CRS is computationally indistinguishable from CRS computed by $\mathsf{TrapGen}(1^\lambda, x, aux)$

– Correctness: Assuming the Correctness does not hold, it means the transcript of one of the repetitions is not accepted. It contradicts Lemma 7.

□

Efficiency analysis of Σ_{NDH} Here we compare the efficiency of our Π_{NDH} with the NIZK protocol obtained by applying the FS transform using a CIHF to the well-known protocol $\Sigma_{\mathsf{DH}} = (\mathsf{Gen}_{\mathsf{DH}}, \mathsf{P}_{\mathsf{DH}}, \mathsf{V}_{\mathsf{DH}})$ used to prove that a tuple is non-DH tuple. We recall how such a protocol works in Figure 4.

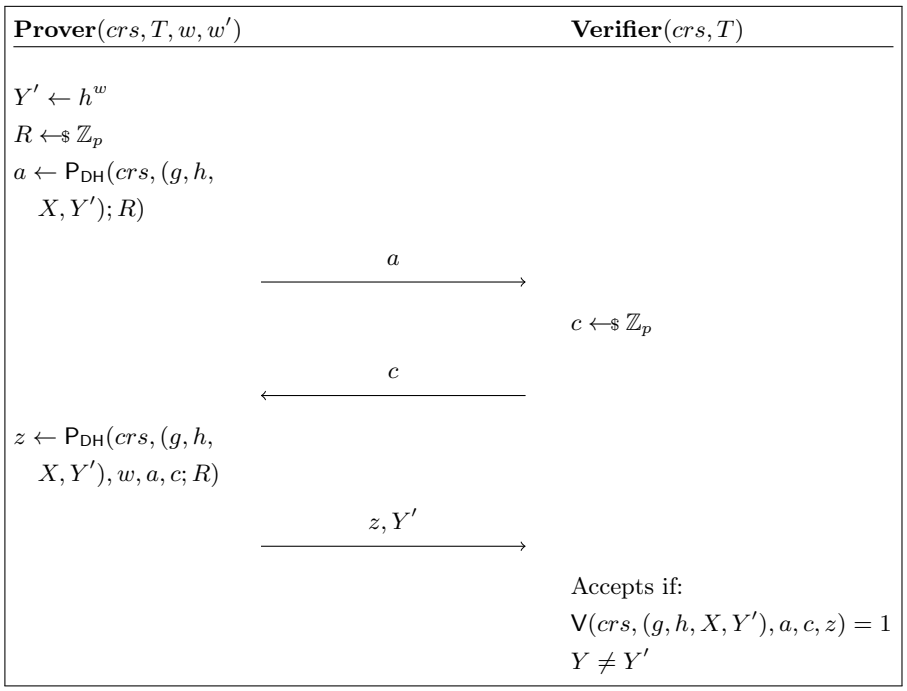

Fig. 4. The protocol for non-DH from DH

Because no expensive operations are introduced in this conversion, the efficiency is the same as Σ_{DH}. Also, the FS transform does not introduce any expensive operations.

Hense, we compare the efficiency of Σ^t_{NDH} with Σ^t_{DH} from Theorem 2:

– Considering the security parameter $\lambda = 2048$, and $\epsilon = \frac{10}{11}$. Then p is 1024 bits.
– Π_{DH}: It requires 1024 repetitions, and in each repetition, the prover needs to compute 2 exponentiations, and the verifier needs to compute 4 exponentiations. In total, it requires 2048 exponentiations for the prover and 4096 exponentiations for the verifier.

– Π_{NDH}: It requires $\frac{1024}{\mathcal{K}\log_2(1024)} = \frac{103}{\mathcal{K}}$ repetitions.
  - If we make $\mathcal{K} = 10$, then the required repetition is 11. In each repetition, the prover needs to compute 4 exponentiations and the verifier needs to compute 4 exponentiations. In total, the prover needs to compute 44 exponentiations and the verifier needs to compute 44 exponentiations.
  - We also want to emphasize that, reducing the number of repetition only influence the reduction of soundness. In the honest execution, neither prover nor verifier does the brute force search computation to get c from h^c.

Also, we can use following formula to get lower bound of λ, s.t. Π_{NDH} more efficient than Π_{DH}, and we consider the total number of exponentiations:

$$
\begin{aligned}
6\lambda^{\epsilon} &\geq 8\frac{\lambda^{\epsilon}}{\mathcal{K}\epsilon\log_2(\lambda)} \\
\log_2(\lambda) &\geq \frac{4}{3\mathcal{K}\epsilon} \\
\lambda &\geq 2^{\frac{4}{3\mathcal{K}\epsilon}}
\end{aligned}
$$

3 NIZK with adaptive multi-theorem ZK

In this section, we show how to obtain our adaptive multi-theorem ZK and sound NIZK protocol for an NP language L, assuming that we have an ID trapdoor sigma-protocol $\Sigma_L = (\mathsf{Gen}_L, \mathsf{P}_L, \mathsf{V}_L)$ for L. For our construction we make use of the following tools:

– A hash family $\mathcal{H}$ that is correlation-intractable for all subexponentially sparse relations that are searchable in time T, which is also programmable.
– The SID trapdoor sigma-protocol $\Sigma_{\mathsf{OR}} = (\mathsf{Gen}_{\mathsf{OR}}, \mathsf{P}_{\mathsf{OR}}, \mathsf{V}_{\mathsf{OR}})$ of Section 2.6 for NP language $L_{\mathsf{OR}} = L \vee L_{\mathsf{DH}} = \{((g,h,x),(X,Y)) \mid (g,h,x) \in S_1 \wedge (X,Y) \in S_2 \wedge (\phi(g,h,X,Y) = 1 \vee x \in L)\}$, where $S_1 = \{(g, g^{\alpha}, x) \in G \times G \times \{0,1\}^{\star} \mid \alpha \in \mathbb{Z}_p\}$, $S_2 = \{(h, h^{\beta}) \in G \times G \mid \beta \in \mathbb{Z}_p\}$, and $\phi(g,h,X,Y) = 1$ if and only if $\exists w \in \mathbb{Z}_p : X = g^w \wedge Y = h^w$. The protocol Σ_{OR} has $2^{-\lambda^{\epsilon}}$ soundness for $\epsilon > 0$. We note that this protocol can be obtained starting from Σ_L and any SID trapdoor sigma protocol Σ_{DH} for L_{DH}. We provide an example (Σ^t_{DH} from Theorem 2) to be used as Σ_{DH}.
– A SID trapdoor sigma-protocol $\Sigma_{\mathsf{NDH}} = (\mathsf{Gen}_{\mathsf{NDH}}, \mathsf{P}_{\mathsf{NDH}}, \mathsf{V}_{\mathsf{NDH}})$ for L_{NDH}. Σ_{NDH} need to have $2^{-\lambda^{\epsilon}}$ soundness for $\epsilon > 0$.

We denote the obtained NIZK protocol with $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$. The Setup algorithm works as follows:

– $crs_L \leftarrow\!\!\$\ \mathsf{Gen}_L(1^{\lambda})$, $crs_{\mathsf{DH}} \leftarrow crs_{\mathsf{NDH}} \leftarrow \emptyset$, $g \leftarrow\!\!\$\ G.\mathsf{Gen}(1^{\lambda})$, $x \leftarrow\!\!\$\ \mathbb{Z}_p, h \leftarrow g^x$, $k \leftarrow\!\!\$\ \mathcal{H}.\mathsf{Gen}(1^{\lambda})$.
– output $(crs_L, crs_{\mathsf{DH}}, crs_{\mathsf{NDH}}, (g,h), k)$

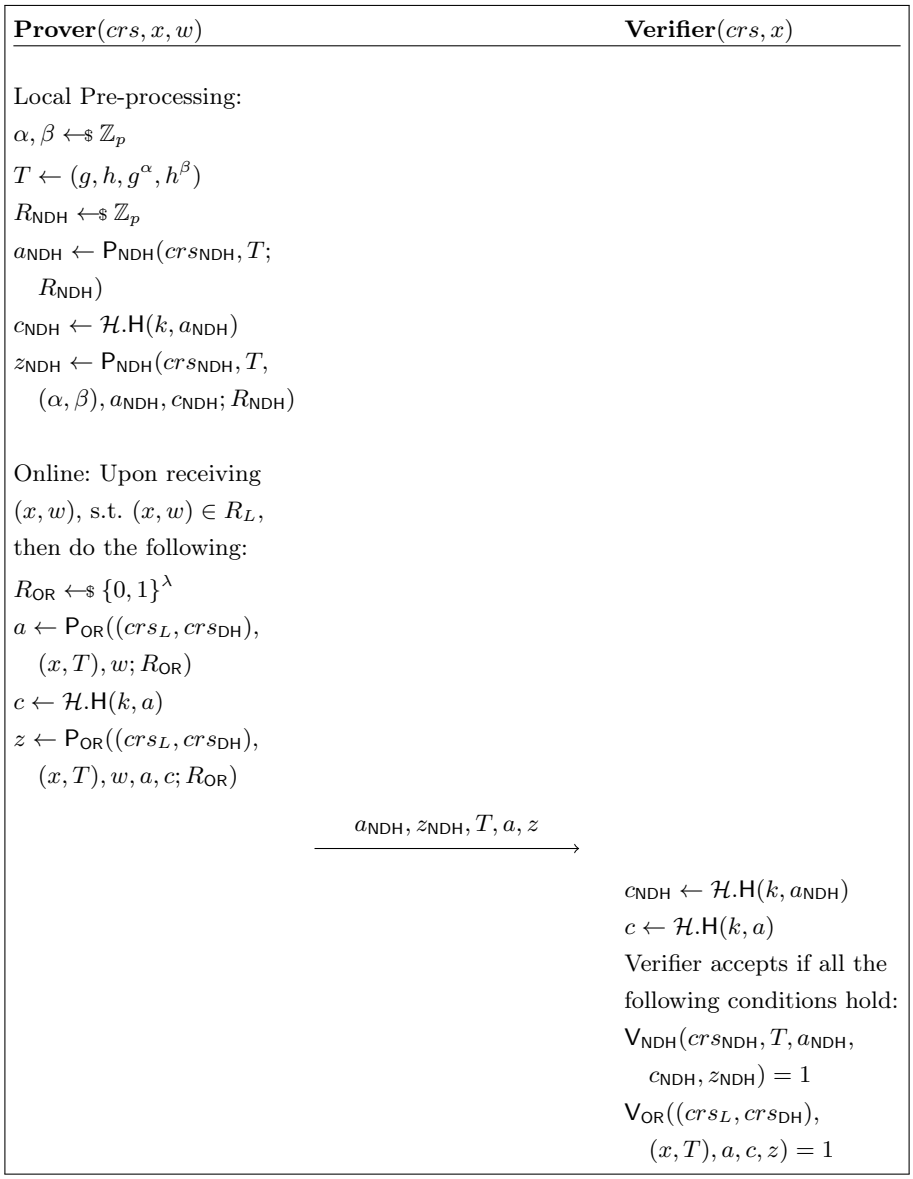

Fig. 5. Our NIZK protocol Π

We formally describe the interaction between the prover and the verifier of Π in Fig. 5.

Before proving the security of Π, we need to prove that the FS transform applied on Σ_{OR} yields a WI semi-adaptive sound non-interactive protocol. This comes immediately from Lemma 3, Lemma 4, and Lemma 5. Hence, if we denote

with Π_{OR} the non-interactive protocol resulting from the application of the FS transform on Σ_{OR} we can claim the following.

Theorem 3. *Π_{OR} is WI semi-adaptive sound for L_{OR}.*

We are now ready to prove our main lemmas.

Lemma 9. *Let Π be the protocol of Fig. 5, then Π is sound.*

Proof. Assuming Π is not sound, then there exists a PPT algorithm $\mathcal{A}$, s.t.:

$$Pr_x\left[x \notin L \wedge \mathsf{V}(crs, x, \pi) = 1 \mid crs \leftarrow\!\$\ \mathsf{Setup}(1^{|x|}, 1^\lambda); \pi \leftarrow\!\$\ \mathcal{A}(crs, x)\right] \geq \delta(\lambda).$$

To make V accept when $x \notin L$, there are 2 possibilities:

- When T is a DH tuple, $\mathsf{V}_{\mathsf{NDH}}(crs_{\mathsf{NDH}}, T, a_{\mathsf{NDH}}, c_{\mathsf{NDH}}, z_{\mathsf{NDH}}) = 1$, and $\mathsf{V}_{\mathsf{OR}}((crs_L, crs_{\mathsf{DH}}), (x, T), a, c, z) = 1$. If $\mathsf{V}_{\mathsf{NDH}}$ accepts when $T \notin L_{\mathsf{NDH}}$ then it means that $\mathcal{A}$ can find $(a_{\mathsf{NDH}}, c_{\mathsf{NDH}}, z_{\mathsf{NDH}})$ to make $(crs_{\mathsf{NDH}}, T, a_{\mathsf{NDH}}, c_{\mathsf{NDH}}, z_{\mathsf{NDH}})$ accepting with non-negligible probability, and it directly contradicts to the semi-adaptive soundness of Π_{NDH}. Formally, we can construct the following adversary $\mathcal{A}'$:

  $\mathcal{A}'(crs_{\mathsf{NDH}}, k, \alpha)$:

  $crs_L \leftarrow\!\$\ \mathsf{Gen}_L(1^\lambda), crs_{\mathsf{DH}} \leftarrow \emptyset$, parsing α as $(g, h), w \leftarrow\!\$\ \mathbb{Z}_p, \beta \leftarrow (g^w, h^w)$,

  $x \leftarrow (g, h, \beta), crs \leftarrow (crs_L, crs_{\mathsf{DH}}, crs_{\mathsf{NDH}}, (g, h), k)$

  waiting for receiving all π from $\mathcal{A}(crs, x)$

  return all (π, β)

  Now we have the following observation: 1) $\mathcal{A}$ works correctly. We know $crs_L \leftarrow\!\$\ \mathsf{Gen}_L(1^\lambda)$, $crs_{\mathsf{DH}} = \emptyset$. Also, the hash key k, crs_{NDH} and (g, h) are provided by the challenger, so we can conclude that crs is the same as $crs \leftarrow\!\$\ \mathsf{Setup}(1^{|x|}, 1^\lambda)$. 2) The output of $\mathcal{A}$ makes V accept with non-negligible probability, and it means that we find an accepting proof π when $(\alpha, \beta) \notin L_{\mathsf{NDH}}$. This contradicts Lemma 5.
- When T is a non-DH tuple, $\mathsf{V}_{\mathsf{NDH}}(crs, T, a_{\mathsf{NDH}}, c_{\mathsf{NDH}}, z_{\mathsf{NDH}}) = 1$, and $\mathsf{V}_{\mathsf{OR}}(crs, (x, T), a, c, z) = 1$. Then $\mathsf{V}_{\mathsf{NDH}}$ accepts because $T \in L_{\mathsf{NDH}}$. However, if V_{OR} accepts when $x \notin L \wedge T \notin L_{\mathsf{DH}}$ it means that the adversary $\mathcal{A}$ is able to find (a, c, z) to make $(crs, (x, T), a, c, z)$ accepting with non-negligible probability, and it directly contradicts the semi-adaptive soundness of Π_{OR}. The reduction is identical to the reduction for Π_{NDH} above, and it contradicts Theorem 3.

We note that in this proof, the security only relies on the soundness of Π_{NDH} and Π_{OR}, where their soundness relies on the CI property of CIHF. We do not use the DDH assumption here. □

Lemma 10. *Let Π be the protocol of Fig. 5, then Π is adaptive multi-theorem zero-knowledge.*

Proof. We have the following simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$, by having SHVZK simulator $\mathsf{Sim}_{\mathsf{NDH}}$ from Σ_{NDH}:

$\mathsf{Sim}(1^{|x|}, 1^\lambda)$:

$crs_L \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda), crs_{\mathsf{DH}} \leftarrow crs_{\mathsf{NDH}} \leftarrow \emptyset, g \leftarrow\!\!\$\ G.\mathsf{Gen}(1^\lambda), x, w_{\mathsf{DH}} \leftarrow\!\!\$\ \mathbb{Z}_p, c_{\mathsf{NDH}} \leftarrow\!\!\$\ \{0,1\}^\lambda$

$h \leftarrow g^x, T_{\mathsf{DH}} \leftarrow (g, h, g^{w_{\mathsf{DH}}}, h^{w_{\mathsf{DH}}}), (a_{\mathsf{NDH}}, z_{\mathsf{NDH}}) \leftarrow \mathsf{Sim}_{\mathsf{NDH}}(T_{\mathsf{DH}}, c_{\mathsf{NDH}})$

$k \leftarrow\!\!\$\ \mathsf{Samp}(1^\lambda, a_{\mathsf{NDH}}, c_{\mathsf{NDH}}), crs \leftarrow (crs_L, crs_{\mathsf{DH}}, crs_{\mathsf{NDH}}, (g, h), k)$

$\tau_{\mathsf{Sim}} \leftarrow (T_{\mathsf{DH}}, w_{\mathsf{DH}}, a_{\mathsf{NDH}}, z_{\mathsf{NDH}})$

return crs, τ_{Sim}

$\mathsf{Sim}(crs, \tau_{\mathsf{Sim}}, x)$:

$R_{\mathsf{OR}} \leftarrow\!\!\$\ \{0,1\}^\lambda, a \leftarrow \mathsf{P}_{\mathsf{OR}}((crs_L, crs_{\mathsf{DH}}), (x, T_{\mathsf{DH}}), w_{\mathsf{DH}}; R_{\mathsf{OR}}), c \leftarrow \mathcal{H}.\mathsf{H}(k, a)$

$z \leftarrow \mathsf{P}_{\mathsf{OR}}((crs_L, crs_{\mathsf{DH}}), (x, T_{\mathsf{DH}}), w_{\mathsf{DH}}, a, c; R_{\mathsf{OR}})$

return $(a_{\mathsf{NDH}}, z_{\mathsf{NDH}}, T_{\mathsf{DH}}, a, z)$

We prove this lemma through hybrid experiments. We denote the output of adversary in the hybrid $\mathcal{H}_i$ with $out^{\mathcal{H}_i}$, where the index $i \in \{0,1,2,3\}$. We want to show for $k = \{0,1,2\}$, for any PPT algorithm $\mathcal{A}$: $|\Pr[\mathcal{A}(out^{\mathcal{H}_k}) = 1] - \Pr[\mathcal{A}(out^{\mathcal{H}_{k+1}}) = 1]| \leq \nu(\lambda)$. We note that $out^{\mathcal{H}_0}$ corresponds to the output of the adversary in the real game, and $out^{\mathcal{H}_4}$ corresponds to the output of the adversary in the simulated experiments. We highlight the part that has differences for better understanding:

$\mathcal{H}_0$:

$State_{\mathcal{A}} \leftarrow \emptyset$

$crs_L \leftarrow\!\!\$\ \mathsf{Gen}_L(1^\lambda), crs_{\mathsf{NDH}} \leftarrow crs_{\mathsf{DH}} \leftarrow \emptyset, g \leftarrow\!\!\$\ G.\mathsf{Gen}(1^\lambda)$

$x, \alpha, \beta, R_{\mathsf{NDH}} \leftarrow\!\!\$\ \mathbb{Z}_p$

$h \leftarrow g^x, k \leftarrow\!\!\$\ \mathcal{H}.\mathsf{Gen}(1^\lambda)$

$crs \leftarrow (crs_L, crs_{\mathsf{DH}}, crs_{\mathsf{NDH}}, (g,h), k)$

$T \leftarrow (g, h, g^\alpha, h^\beta)$

$a_{\mathsf{NDH}} \leftarrow \mathsf{P}_{\mathsf{NDH}}(crs_{\mathsf{NDH}}, T; R_{\mathsf{NDH}})$

$c_{\mathsf{NDH}} \leftarrow \mathcal{H}.\mathsf{H}(k, a_{\mathsf{NDH}})$

$z_{\mathsf{NDH}} \leftarrow \mathsf{P}_{\mathsf{NDH}}(crs_{\mathsf{NDH}}, T, (\alpha, \beta), a_{\mathsf{NDH}}, c_{\mathsf{NDH}}; R_{\mathsf{NDH}})$; repeat

$(State_{\mathcal{A}}, x, w) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, crs, State_{\mathcal{A}})$

if $(x,w) \in R_L$ then $R_{\mathsf{OR}} \leftarrow\!\!\$\ \{0,1\}^\lambda, a \leftarrow \mathsf{P}_{\mathsf{OR}}((crs_L, crs_{\mathsf{DH}}), (x,T), w; R_{\mathsf{OR}})$

$c \leftarrow \mathcal{H}.\mathsf{H}(k,a), z \leftarrow \mathsf{P}_{\mathsf{OR}}((crs_L, crs_{\mathsf{DH}}), (x,T), w, a, c; R_{\mathsf{OR}})$

$\pi \leftarrow (a_{\mathsf{NDH}}, z_{\mathsf{NDH}}, T, a, z)$

else $\pi \leftarrow \emptyset$

$(State_{\mathcal{A}}, cont, d) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, State_{\mathcal{A}}, \pi)$

until $cont = \mathsf{false}$

return $d = 0$

$\mathcal{H}_1 : State_{\mathcal{A}} \leftarrow \emptyset, crs_L \leftarrow\!\!\$\ \mathsf{Gen}_L(1^\lambda), crs_{\mathsf{NDH}} \leftarrow crs_{\mathsf{DH}} \leftarrow \emptyset, g \leftarrow\!\!\$\ G.\mathsf{Gen}(1^\lambda)$

$x, \alpha, \beta, R_{\mathsf{NDH}} \leftarrow\!\!\$\ \mathbb{Z}_p, c_{\mathsf{NDH}} \leftarrow\!\!\$\ \{0,1\}^\lambda, h \leftarrow g^x, T \leftarrow (g, h, g^\alpha, h^\beta)$

$a_{\mathsf{NDH}} \leftarrow \mathsf{P}_{\mathsf{NDH}}(crs_{\mathsf{NDH}}, T; R_{\mathsf{NDH}}),$ $k \leftarrow \mathsf{Samp}(1^\lambda, a_{\mathsf{NDH}}, c_{\mathsf{NDH}})$

$crs \leftarrow (crs_L, crs_{\mathsf{DH}}, crs_{\mathsf{NDH}}, (g, h), k), z_{\mathsf{NDH}} \leftarrow \mathsf{P}_{\mathsf{NDH}}(crs_{\mathsf{NDH}}, T, (\alpha, \beta), a_{\mathsf{NDH}}, c_{\mathsf{NDH}}; R_{\mathsf{NDH}})$

repeat

$(State_{\mathcal{A}}, x, w) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, crs, State_{\mathcal{A}})$

if $(x, w) \in R_L$ then $R_{\mathsf{OR}} \leftarrow\!\!\$\ \{0,1\}^\lambda, a \leftarrow \mathsf{P}_{\mathsf{OR}}((crs_L, crs_{\mathsf{DH}}), (x, T), w; R_{\mathsf{OR}})$

$c \leftarrow \mathcal{H}.\mathsf{H}(k, a), z \leftarrow \mathsf{P}_{\mathsf{OR}}((crs_L, crs_{\mathsf{DH}}), (x, T), w, a, c; R_{\mathsf{OR}})$

$\pi \leftarrow (a_{\mathsf{NDH}}, z_{\mathsf{NDH}}, T, a, z)$

else $\pi \leftarrow \emptyset$

$(State_{\mathcal{A}}, cont, d) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, State_{\mathcal{A}}, \pi)$

until $cont = \mathsf{false}$

return $d = 0$

$\mathcal{H}_2 : State_{\mathcal{A}} \leftarrow \emptyset, crs_L \leftarrow\!\!\$\ \mathsf{Gen}_L(1^\lambda), crs_{\mathsf{NDH}} \leftarrow crs_{\mathsf{DH}} \leftarrow \emptyset, g \leftarrow\!\!\$\ G.\mathsf{Gen}(1^\lambda)$

$x, \alpha, \beta, R_{\mathsf{NDH}} \leftarrow\!\!\$\ \mathbb{Z}_p, c_{\mathsf{NDH}} \leftarrow\!\!\$\ \{0,1\}^\lambda, h \leftarrow g^x, T \leftarrow (g, h, g^\alpha, h^\beta)$

$(a_{\mathsf{NDH}}, z_{\mathsf{NDH}}) \leftarrow \mathsf{Sim}_{\mathsf{NDH}}(T, c_{\mathsf{NDH}}), k \leftarrow \mathsf{Samp}(1^\lambda, a_{\mathsf{NDH}}, c_{\mathsf{NDH}})$

$crs \leftarrow (crs_L, crs_{\mathsf{DH}}, crs_{\mathsf{NDH}}, (g, h), k)$; repeat

$(State_{\mathcal{A}}, x, w) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, crs, State_{\mathcal{A}})$

if $(x, w) \in R_L$ then $R_{\mathsf{OR}} \leftarrow\!\!\$\ \{0,1\}^\lambda, a \leftarrow \mathsf{P}_{\mathsf{OR}}((crs_L, crs_{\mathsf{DH}}), (x, T), w; R_{\mathsf{OR}})$

$c \leftarrow \mathcal{H}.\mathsf{H}(k, a), z \leftarrow \mathsf{P}_{\mathsf{OR}}((crs_L, crs_{\mathsf{DH}}), (x, T), w, a, c; R_{\mathsf{OR}})$

$\pi \leftarrow (a_{\mathsf{NDH}}, z_{\mathsf{NDH}}, T, a, z)$

else $\pi \leftarrow \emptyset$

$(State_{\mathcal{A}}, cont, d) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, State_{\mathcal{A}}, \pi)$

until $cont = \mathsf{false}$

return $d = 0$

$\mathcal{H}_3 : State_{\mathcal{A}} \leftarrow \emptyset, crs_L \leftarrow\!\!\$\ \mathsf{Gen}_L(1^\lambda), crs_{\mathsf{NDH}} \leftarrow crs_{\mathsf{DH}} \leftarrow \emptyset, g \leftarrow\!\!\$\ G.\mathsf{Gen}(1^\lambda)$

$x, w_{\mathsf{DH}}, R_{\mathsf{NDH}} \leftarrow\!\!\$\ \mathbb{Z}_p, c_{\mathsf{NDH}} \leftarrow\!\!\$\ \{0,1\}^\lambda, h \leftarrow g^x,$ $T_{\mathsf{DH}} \leftarrow (g, h, g^{w_{\mathsf{DH}}}, h^{w_{\mathsf{DH}}})$

$(a_{\mathsf{NDH}}, z_{\mathsf{NDH}}) \leftarrow \mathsf{Sim}_{\mathsf{NDH}}(T_{\mathsf{DH}}, c_{\mathsf{NDH}}), k \leftarrow \mathsf{Samp}(1^\lambda, a_{\mathsf{NDH}}, c_{\mathsf{NDH}})$

$crs \leftarrow (crs_L, crs_{\mathsf{DH}}, crs_{\mathsf{NDH}}, (g, h), k)$; repeat

$(State_{\mathcal{A}}, x, w) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, crs, State_{\mathcal{A}})$

if $(x, w) \in R_L$ then $R_{\mathsf{OR}} \leftarrow\!\!\$\ \{0,1\}^\lambda, a \leftarrow \mathsf{P}_{\mathsf{OR}}((crs_L, crs_{\mathsf{DH}}), (x, T_{\mathsf{DH}}), w; R_{\mathsf{OR}})$

$c \leftarrow \mathcal{H}.\mathsf{H}(k, a), z \leftarrow \mathsf{P}_{\mathsf{OR}}((crs_L, crs_{\mathsf{DH}}), (x, T_{\mathsf{DH}}), w, a, c; R_{\mathsf{OR}})$

$\pi \leftarrow (a_{\mathsf{NDH}}, z_{\mathsf{NDH}}, T_{\mathsf{DH}}, a, z)$

else $\pi \leftarrow \emptyset$

$(State_{\mathcal{A}}, cont, d) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, State_{\mathcal{A}}, \pi)$

until $cont = \mathsf{false}$

return $d = 0$

$\mathcal{H}_4$: $State_{\mathcal{A}} \leftarrow \emptyset, crs_L \leftarrow\!\!\$\ \mathsf{Gen}_L(1^\lambda), crs_{\mathsf{NDH}} \leftarrow crs_{\mathsf{DH}} \leftarrow \emptyset, g \leftarrow\!\!\$\ G.\mathsf{Gen}(1^\lambda)$
$x, w_{\mathsf{DH}}, R_{\mathsf{NDH}} \leftarrow\!\!\$\ \mathbb{Z}_p, c_{\mathsf{NDH}} \leftarrow\!\!\$\ \{0,1\}^\lambda, h \leftarrow g^x, T_{\mathsf{DH}} \leftarrow (g, h, g^{w_{\mathsf{DH}}}, h^{w_{\mathsf{DH}}})$
$(a_{\mathsf{NDH}}, z_{\mathsf{NDH}}) \leftarrow \mathsf{Sim}_{\mathsf{NDH}}(T_{\mathsf{DH}}, c_{\mathsf{NDH}}), k \leftarrow \mathsf{Samp}(1^\lambda, a_{\mathsf{NDH}}, c_{\mathsf{NDH}})$
  $crs \leftarrow (crs_L, crs_{\mathsf{DH}}, crs_{\mathsf{NDH}}, (g,h), k)$; repeat
    $(State_{\mathcal{A}}, x, w) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, crs, State_{\mathcal{A}})$
    if $(x,w) \in R_L$ then $R_{\mathsf{OR}} \leftarrow\!\!\$\ \{0,1\}^\lambda$, $a \leftarrow \mathsf{P}_{\mathsf{OR}}((crs_L, crs_{\mathsf{DH}}), (x, T_{\mathsf{DH}}), w_{\mathsf{DH}}; R_{\mathsf{OR}})$
      $c \leftarrow \mathcal{H}.\mathsf{H}(k, a)$, $z \leftarrow \mathsf{P}_{\mathsf{OR}}((crs_L, crs_{\mathsf{DH}}), (x, T_{\mathsf{DH}}), w_{\mathsf{DH}}, a, c; R_{\mathsf{OR}})$
      $\pi \leftarrow (a_{\mathsf{NDH}}, z_{\mathsf{NDH}}, T_{\mathsf{DH}}, a, z)$
    else $\pi \leftarrow \emptyset$
    $(State_{\mathcal{A}}, cont, d) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, State_{\mathcal{A}}, \pi)$
  until $cont = \mathsf{false}$
return $d = 0$

Then we have the following reductions:

- **Reduction** 1: Assuming there exists a PPT algorithm $\mathcal{A}$ that $\left|\Pr\left[\mathcal{A}(out^{\mathcal{H}_0}) = 1\right] - \Pr\left[\mathcal{A}(out^{\mathcal{H}_1}) = 1\right]\right| \geq \delta(\lambda)$, then we can construct an adversary $\mathcal{A}'$ that can break the programmability of CIHF $\mathcal{H}$ (Definition 14) through the following reduction:
  - $\mathcal{A}'$ queries the challenger of the programmability of $\mathcal{H}$ that sends back the hash key k
  - $\mathcal{A}'$ samples crs_L by using Gen_L, samples crs_{DH}, crs_{NDH}, (g,h) correspondingly, and does $crs \leftarrow (crs_L, crs_{\mathsf{DH}}, crs_{\mathsf{NDH}}, (g,h), k)$
  - $\mathcal{A}'$ sends crs to $\mathcal{A}$ to get (x,w)
  - $\mathcal{A}'$ preparing π by using (x,w) and sends it to $\mathcal{A}$
  - $\mathcal{A}'$ outputs the output of $\mathcal{A}$

  We now observe that if the challenger uses $\mathcal{H}.\mathsf{Gen}$ to sample k, we are in $\mathcal{H}_0$, otherwise, we are in $\mathcal{H}_1$. This implies $\mathcal{H}_0 \approx \mathcal{H}_1$.
- **Reduction** 2: Assuming there exists a PPT algorithm $\mathcal{A}$ that $\left|\Pr\left[\mathcal{A}(out^{\mathcal{H}_1}) = 1\right] - \Pr\left[\mathcal{A}(out^{\mathcal{H}_2}) = 1\right]\right| \geq \delta(\lambda)$, then we can construct an adversary $\mathcal{A}'$ that can break the SHVZK of Σ_{NDH} through the following reduction:
  - $\mathcal{A}'$ queries the challenger of the SHVZK of Σ_{NDH} that sends back the proof $a_{\mathsf{NDH}}, z_{\mathsf{NDH}}$
  - $\mathcal{A}'$ samples crs_L by using Gen_L, samples k by using Samp, samples crs_{DH}, crs_{NDH}, (g,h) correspondingly, and does $crs \leftarrow (crs_L, crs_{\mathsf{DH}}, crs_{\mathsf{NDH}}, (g,h), k)$
  - $\mathcal{A}'$ sends crs to $\mathcal{A}$ to get (x,w)
  - $\mathcal{A}'$ preparing π_{OR} by using (x,w) and does $\pi \leftarrow (\pi_{\mathsf{OR}}, T, a_{\mathsf{NDH}}, z_{\mathsf{NDH}})$
  - $\mathcal{A}'$ sends π to $\mathcal{A}$, and outputs the output of $\mathcal{A}$

We now observe that if the challenger provides a real transcript, we are in $\mathcal{H}_1$, otherwise, we are in $\mathcal{H}_2$. This implies $\mathcal{H}_1 \approx \mathcal{H}_2$.

– **Reduction** 3: Assuming there exists a PPT algorithm $\mathcal{A}$ that $\left|\Pr\left[\mathcal{A}(out^{\mathcal{H}_2}) = 1\right] - \Pr\left[\mathcal{A}(out^{\mathcal{H}_3}) = 1\right]\right| \geq \delta(\lambda)$, then we can construct an adversary $\mathcal{A}'$ that can break the DDH hardness assumption (Definition 1) through the following reduction:

- $\mathcal{A}'$ queries the challenger of the DDH hardness assumption that sends back the tuple $T = (g, h, X, Y)$
- $\mathcal{A}'$ gets (g, h) from T, and uses T to generate $(a_{\mathsf{NDH}}, z_{\mathsf{NDH}})$ from using $\mathsf{Sim}_{\mathsf{NDH}}$
- $\mathcal{A}'$ samples crs_L by using Gen_L, samples k by using Samp, samples crs_{DH}, crs_{NDH} correspondingly, and does $crs \leftarrow (crs_L, crs_{\mathsf{DH}}, crs_{\mathsf{NDH}}, (g, h), k)$
- $\mathcal{A}'$ sends crs to $\mathcal{A}$ to get (x, w)
- $\mathcal{A}'$ preparing π_{OR} by using (x, w) and does $\pi \leftarrow (\pi_{\mathsf{OR}}, T, a_{\mathsf{NDH}}, z_{\mathsf{NDH}})$
- $\mathcal{A}'$ sends π to $\mathcal{A}$, and outputs the output of $\mathcal{A}$

We now observe that if the challenger provides a non-DH tuple, we are in $\mathcal{H}_2$, otherwise, we are in $\mathcal{H}_3$. This implies $\mathcal{H}_2 \approx \mathcal{H}_3$.

– **Reduction** 4: Assuming there exists a PPT algorithm $\mathcal{A}$ that $\left|\Pr\left[\mathcal{A}(out^{\mathcal{H}_3}) = 1\right] - \Pr\left[\mathcal{A}(out^{\mathcal{H}_4}) = 1\right]\right| \geq \delta(\lambda)$, then we can construct an adversary $\mathcal{A}'$ that can break the WI of Π_{OR} through the following reduction:

- $\mathcal{A}'$ queries the challenger of the WI of Π_{OR} that sends back $x, \pi_{\mathsf{OR}} = (a_{\mathsf{OR}}, z_{\mathsf{OR}})$
- $\mathcal{A}'$ samples crs_L by using Gen_L, samples k by using Samp, samples crs_{DH}, crs_{NDH}, (g, h) correspondingly, and does $crs \leftarrow (crs_L, crs_{\mathsf{DH}}, crs_{\mathsf{NDH}}, (g, h), k)$
- $\mathcal{A}'$ sends crs to $\mathcal{A}$ to get (x, w)
- $\mathcal{A}'$ use π_{OR} from the challenger, and does $\pi \leftarrow (\pi_{\mathsf{OR}}, T_{\mathsf{DH}}, a_{\mathsf{NDH}}, z_{\mathsf{NDH}})$
- $\mathcal{A}'$ sends π to $\mathcal{A}$, and outputs the output of $\mathcal{A}$

We now observe that if the challenger provide Π_{OR} by using w, where $(x, w) \in R_L$, we are in $\mathcal{H}_3$, otherwise we are in $\mathcal{H}_4$. This implies $\mathcal{H}_3 \approx \mathcal{H}_4$.
We can concludes $\mathcal{H}_0 \approx \mathcal{H}_1 \approx \mathcal{H}_2 \approx \mathcal{H}_3 \approx \mathcal{H}_4$. Therefore, the simulated transcript from Sim is computationally indistinguishable from a transcript in the real game.

□

On the adaptive soundness of our protocol. In the previous section, we showed that Π is (non-adaptive) sound and adaptive multi-theorem ZK. In this section, we argue that it is possible to slightly modify Π and get a protocol that enjoys the same properties as Π, but in addition, it is also adaptive-sound.

In [CPV20] the authors show that if the input of the hash function used in the FS transform contains also the theorem (and not just the first round of

the underlying protocol), and moreover the trapdoor sigma-protocol is *instance-independent* then the resulting NIZK is adaptive sound. As an additional contribution, the authors of [CPV20] show that any sigma-protocol can be turned into an instance-independent trapdoor sigma-protocol (this construction has an overhead, that requires computing two ciphertexts for each bit of the challenge of the starting trapdoor sigma-protocol).

Hence, using the results of [CPV20], we can construct an instance-independent trapdoor sigma-protocol for the language $L \vee L_{\mathsf{DH}}$. If we apply the FS transform using as the input of the hash-function also x then the final NIZK protocol we obtain is both adaptive sound and adaptive multi-theorem ZK.

References

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988.

[BKM20] Zvika Brakerski, Venkata Koppula, and Tamer Mour. NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 738–767. Springer, Heidelberg, August 2020.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019.

[CCR16] Ran Canetti, Yilei Chen, and Leonid Reyzin. On the correlation intractability of obfuscated pseudorandom functions. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 389–415. Springer, Heidelberg, January 2016.

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 91–122. Springer, Heidelberg, April / May 2018.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, August 1994.

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Heidelberg, August 1993.

[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 Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 83–111. Springer, Heidelberg, January 2016.

[CPV20] Michele Ciampi, Roberto Parisella, and Daniele Venturi. On adaptive security of delayed-input sigma protocols and fiat-shamir NIZKs. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20*, vol-

ume 12238 of *LNCS*, pages 670–690. Springer, Heidelberg, September 2020.

[CSW20] Ran Canetti, Pratik Sarkar, and Xiao Wang. Triply adaptive UC NIZK. Cryptology ePrint Archive, Report 2020/1212, 2020. https://eprint.iacr.org/2020/1212.

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

[DMP88] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 52–72. Springer, Heidelberg, August 1988.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Heidelberg, December 2012.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st FOCS*, pages 308–317. IEEE Computer Society Press, October 1990.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[HL18] Justin Holmgren and Alex Lombardi. Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In Mikkel Thorup, editor, *59th FOCS*, pages 850–858. IEEE Computer Society Press, October 2018.

[KRR17] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of Fiat-Shamir for proofs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 224–251. Springer, Heidelberg, August 2017.

[LNPY22] Benoît Libert, Khoa Nguyen, Thomas Peters, and Moti Yung. One-shot fiat-shamir-based NIZK arguments of composite residuosity and logarithmic-size ring signatures in the standard model. In *EUROCRYPT 2022, Part II*, LNCS, pages 488–519. Springer, Heidelberg, June 2022.

[LS91] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 353–365. Springer, Heidelberg, August 1991.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 89–114. Springer, Heidelberg, August 2019.

[YZ06] Moti Yung and Yunlei Zhao. Interactive zero-knowledge with restricted random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 21–40. Springer, Heidelberg, March 2006.