On Adaptive Security of Delayed-Input Sigma Protocols and Fiat-Shamir NIZKs

Michele Ciampi [*1], Roberto Parisella [†2], and Daniele Venturi [‡3]

[1] *The University of Edinburgh, UK*
[2] *Simula UiB, Norway*
[3] *Sapienza University of Rome, Italy*

Abstract

We study *adaptive security* of delayed-input Sigma protocols and non-interactive zero-knowledge (NIZK) proof systems in the common reference string (CRS) model. Our contributions are threefold:

- We exhibit a generic compiler taking any delayed-input Sigma protocol and returning a delayed-input Sigma protocol satisfying *adaptive-input* special honest-verifier zero-knowledge (SHVZK). In case the initial Sigma protocol also satisfies *adaptive-input* special soundness, our compiler preserves this property.
- We revisit the recent paradigm by Canetti *et al.* (STOC 2019) for obtaining NIZK proof systems in the CRS model via the Fiat-Shamir transform applied to so-called *trapdoor* Sigma protocols, in the context of adaptive security. In particular, assuming correlation-intractable hash functions for all sparse relations, we prove that Fiat-Shamir NIZKs satisfy either:
  (i) Adaptive soundness (and non-adaptive zero-knowledge), so long as the challenge is obtained by hashing both the prover's first round and the instance being proven;
  (ii) Adaptive zero-knowledge (and non-adaptive soundness), so long as the challenge is obtained by hashing only the prover's first round, and further assuming that the initial trapdoor Sigma protocol satisfies adaptive-input SHVZK.
- We exhibit a generic compiler taking any Sigma protocol and returning a *trapdoor* Sigma protocol. Unfortunately, this transform does not preserve the delayed-input property of the initial Sigma protocol (if any). To complement this result, we also give yet another compiler taking any delayed-input trapdoor Sigma protocol and returning a delayed-input trapdoor Sigma protocol with adaptive-input SHVZK.

An attractive feature of our first two compilers is that they allow obtaining *efficient* delayed-input Sigma protocols with adaptive security, and *efficient* Fiat-Shamir NIZKs with adaptive soundness (and non-adaptive zero knowledge) in the CRS model. Prior to our work, the latter was only possible using generic NP reductions.

Keywords. Sigma protocols; Non-interactive zero knowledge; Adaptive security.

[*] mciampi@ed.ac.uk
[†] roberto@simula.no
[‡] venturi@di.uniroma1.it

Contents

1 Introduction **1**
- 1.1 Our Contributions . . . 3
- 1.2 Technical Overview . . . 3
  - 1.2.1 Adaptive-input SHVZK . . . 3
  - 1.2.2 Adaptive Security of Fiat-Shamir NIZKs . . . 4
- 1.3 Applications . . . 6

2 Preliminaries **7**
- 2.1 Standard Definitions . . . 7
- 2.2 Public-Key Encryption . . . 7
- 2.3 Sigma Protocols . . . 8
- 2.4 The DDH Assumption . . . 8
- 2.5 Instance-dependent Trapdoor Commitment . . . 9
- 2.6 Correlation Intractable Hash Families . . . 10
- 2.7 Non-Interactive Argument Systems . . . 11

3 A Compiler for Adaptive-input HVZK **12**
- 3.1 Delayed-input Sigma Protocols . . . 12
- 3.2 The Transformation . . . 13

4 Adaptive Security of the Fiat-Shamir Transform **17**
- 4.1 Trapdoor Sigma Protocols and the Fiat-Shamir Transform . . . 17
- 4.2 From CRSigma Protocols to Trapdoor Sigma Protocols . . . 19

5 Adding Adaptive-Input SHVZK **21**
- 5.1 The Sigma Protocol for DH Tuples . . . 21
- 5.2 Extractable IDTCs . . . 22
  - 5.2.1 The Basic Construction . . . 23
  - 5.2.2 Extending the Message Space . . . 24
  - 5.2.3 Sigma Protocols for Extractable IDTCs . . . 24
- 5.3 Trapdoor Sigma Protocols with Adaptive SHVZK . . . 25

6 Conclusions **27**

7 Acknowledgments **27**

1 Introduction

Sigma protocols are a special class of three-round public-coin interactive proofs between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, where $\mathcal{P}$'s goal is to convince $\mathcal{V}$ that a common statement x belongs to a given NP language L. The prover knows a witness w (corresponding to x) as auxiliary input, and starts the interaction by sending a first message a (possibly depending on both x, w); the verifier then sends a uniformly random ℓ-bit challenge c, to which the prover replies with a last message z. Finally, the verifier decides whether $x \in L$ based on x and the transcript (a, c, z). Despite *completeness* (*i.e.*, the honest prover always convinces the honest verifier about true statements), Sigma protocols satisfy two additional properties known as *special soundness* (SS) and *special honest-verifier zero knowledge* (SHVZK). The former is a strong form of *soundness* (*i.e.*, no malicious prover can convince the verifier about the veracity of *false* statements $x \notin L$), which in fact implies that Sigma protocols are proofs of knowledge [GMR85, BG93]; the latter requires the existence of an efficient simulator $\mathcal{S}$ that, given any *true* statement $x \in L$ and any possible challenge c, is able to simulate an *honest* transcript (a, c, z) between the prover and the verifier, which in particular implies that honest transcripts do not reveal anything about the witness to the eyes of an honest-but-curious verifier. While Sigma protocols exist for all of NP (as Blum's protocol [Blu86] for Hamiltonian Graphs is a Sigma protocol), the latter comes at the price of expensive NP reductions. Luckily, Sigma protocols also exist for many concrete languages based on number theory and lattices (such as Quadratic Residuosity [GMR85], Discrete Log [Sch90, Oka93], Factoring [FF02], and Learning with Errors [Ste94, Lyu09, AJL+12]), and these protocols are very efficient, thus opening the way to a plethora of cryptographic applications, *e.g.* to constructing different kinds of commitment schemes [DN02, CV05, HL10, Dam10, Lin15, CPSV16] and trapdoor hash functions [BR14], and for obtaining non-interactive zero-knowledge (NIZK) proofs and digital signatures via the celebrated Fiat-Shamir transform [FS87, BR93, PS96]. In this paper we study *adaptive security* for both Sigma protocols and Fiat-Shamir NIZKs.

Delayed-input Sigma protocols. The classical Sigma protocol by Feige, Lapidot and Shamir [FLS90, LS91] for Graph Hamiltonicity (henceforth denoted by FLS) has the special property that the prover can compute the first round of the proof without knowing the graph, so long as it knows the number of vertices ahead of time. In particular, the graph and the corresponding Hamiltonian cycle are only needed to compute the prover's last round. More generally, a Sigma protocol is called *delayed-input* if the prover's first round can be computed given only $n = |x|$ (and without knowing x, w). For such Sigma protocols, the standard definitions of SS and SHVZK may not be sufficient as they do not take into account attackers choosing the statement x adaptively based on a partial transcript (a, c). This limitation may have a negative impact[1] on the applications of delayed-input Sigma protocols, particularly in settings where adaptive security is required. While, the FLS protocol already satisfies both *adaptive-input* SS and *adaptive-input* SHVZK,[2] the latter is only of theoretical interest. Partially motivated by this shortcoming, Ciampi *et al.* [CPS+16] proposed a general transformation for turning any delayed-input Sigma protocol into one satisfying *adaptive-input* SS. This leaves the following open problem.

Q1: *"Do there exist efficient delayed-input Sigma protocols with adaptive security (i.e., satisfying both adaptive-input SS and adaptive-input SHVZK)?"*

[1] We discuss practical applications where adaptive security is of concern in Section 1.3.

[2] Intuitively, adaptive-input SS guarantees extraction even for transcripts (a, c, z) and (a, c', z') for different (possibly adaptively chosen) statements. Similarly, adaptive-input SHVZK requires the simulator to fake the prover's first message given only $n = |x|$.

Fiat-Shamir NIZKs. The Fiat-Shamir transform [FS87] allows to turn a Sigma protocol into a non-interactive proof system by means of a hash functions h with ℓ-bit output. The idea is for the prover to compute a and z as prescribed by the Sigma protocol, where the challenge c is set to $c := h(a||x)$. One can show that this yields a secure NIZK starting from any Sigma protocol, so long as the hash function h is modelled as a random oracle [BR93, FKMV12]. Whether security of the Fiat-Shamir transform can be proven without resorting to random oracles has been a question subject of intensive study. Here, the goal is to instantiate the random oracle with a set of efficiently computable hash functions $\mathcal{H} = \{h_k\}$, where the hash key k is made available to all parties in the form of a common reference string (CRS). Unfortunately, several negative results are known in this respect [DNRS99, Bar01, GK03, BDG+13], which, however, only exclude the possibility of instantiating the Fiat-Shamir transform starting with *any* Sigma protocol or via black-box reductions to falsifiable assumptions. Indeed, a recent line of research[3] established the above negative results can be circumvented:

- Assuming the initial interactive protocol is a *trapdoor* Sigma protocols [CCH+19]. Informally, a trapdoor Sigma protocol is a special Sigma protocol in the CRS model satisfying the following two properties: (i) If the statement x is false, then for every first message a, there is a unique challenge c for which there is an accepting third message z that results in an accepting transcript (a, c, z); (ii) There is a trapdoor associated with the CRS that allows us to efficiently compute this "bad challenge" c from the first message a and the statement x being proven.
- Assuming that $\mathcal{H}$ is a family of correlation-intractable (CI) hash functions [CGH98]. Informally, a family $\mathcal{H}$ satisfies CI w.r.t. some relation R if no efficient attacker given the hash key k can produce an input x such that $(x, h_k(x)) \in R$. CI hash functions w.r.t. broad-enough[4] relations have recently been constructed from a variety of assumptions including program obfuscation [CCR16, MV16, KRR17], strong one-way functions [HL18], key-dependent message secure encryption [CCRR18], circularly-secure fully-homomorphic encryption [CCH+19], LWE [PS19], and LPN along with DDH/QR/DCR/LWE [BKM20].

A natural question is whether Fiat-Shamir NIZKs obtained via CI hash functions are adaptively secure, *i.e.* whether the non-interactive proof resulting from applying the Fiat-Shamir transform to a trapdoor Sigma protocol satisfies both *adaptive* soundness and *adaptive* zero-knowledge in the CRS model.[5] Canetti *et al.* [CCH+19] proved that a slight variant of the FLS protocol directly achieves adaptive security, however, in order to be used in applications, the latter requires expensive NP reductions, and thus results in very inefficient NIZKs. They also provide an efficient instantiation using the classical Sigma protocol for Quadratic Residuosity [GMR85], and more in general starting with any *instance-dependent* trapdoor Sigma protocol (in which the trapdoor is allowed to depend on the statement being proven). Unfortunately, instance-dependent trapdoor Sigma protocols are not sufficient to prove adaptive security of Fiat-Shamir NIZKs, thus leaving the following intriguing open question.

Q2: *"Do there exist efficient trapdoor Sigma protocols allowing to obtain Fiat-Shamir NIZKs with adaptive security (i.e., satisfying both adaptive soundness and adaptive zero knowledge in the CRS model)?"*

[3] This research extends previous results showing that CI is sufficient for proving *soundness* of the Fiat-Shamir transform [DNRS99, BLV03, HMR08].

[4] In particular, sufficient for proving security of Fiat-Shamir NIZKs without random oracles.

[5] The former means that no malicious prover, given the CRS, can produce a false statement along with an accepting non-interactive proof. The latter means that no malicious verifier, given the CRS, can produce a true statement, along with the corresponding witness, for which a non-interactive proof cannot be simulated in polynomial time given the statement alone.

1.1 Our Contributions

In this work, we make progress towards answering the above two open questions in the affirmative. Our first contribution is a general compiler taking any delayed-input Sigma protocol and outputting another delayed-input Sigma protocol (for the same language) with adaptive-input SHVZK. Furthermore, assuming the initial Sigma protocol already satisfies adaptive-input SS, so does the Sigma protocol produced by our compiler. Hence, using the transformation by Ciampi *et al.* [CPS+16], we obtain a general compiler which allows to turn any delayed-input Sigma protocol into one with adaptive security, which is a positive answer to **Q1**. Next, we revisit the framework for obtaining adaptively-secure NIZKs via the Fiat-Shamir transform using CI hash functions. In particular, we show the following two results:

- In case the challenge c is obtained by hashing both the prover's first round a and the statement x (*i.e.,* $c = h_k(a||x)$), trapdoor Sigma protocols are sufficient for proving *adaptive soundness* and *non-adaptive zero knowledge* of Fiat-Shamir NIZKs in the CRS model.
- In case the challenge c is obtained by hashing only the prover's first round a (*i.e.,* $c = h_k(a)$), trapdoor Sigma protocols satisfying soundness (which in turn follows by SS) and *adaptive-input* SHVZK are sufficient for proving *non-adaptive soundness* and *adaptive zero knowledge* of Fiat-Shamir NIZKs in the CRS model.

The fact that hashing both the prover's first message and the statement is essential for obtaining adaptive soundness was already known for the random oracle model [BPW12]. In this vein, our paper confirms this to be sufficient in the plain model as well. Our second contribution is a compiler taking any Sigma protocol and outputting a *trapdoor* Sigma protocol (for the same language). Unfortunately, this compiler does not preserve the delayed-input property of the initial Sigma protocol (if any), and thus, by our result from above, only implies Fiat-Shamir NIZKs with adaptive soundness (but not adaptive zero knowledge). This result can still be interpreted as a partial (positive) answer to **Q2**, as it allows to obtain *efficient* Fiat-Shamir NIZKs with *adaptive soundness* (and non-adaptive zero knowledge) in the CRS model for any language admitting a Sigma protocol. Previously to our work, the latter was possible only using expensive NP reductions. Finally, we also show that any delayed-input trapdoor Sigma protocol can be turned into a delayed-input trapdoor Sigma protocol with adaptive-input SHVZK, which (again by our generalization of [CCH+19]) would be sufficient for obtaining Fiat-Shamir NIZKs with adaptive zero knowledge (and non-adaptive soundness) in the CRS model. Unfortunately, the only example we know of a delayed-input trapdoor Sigma protocol is FLS (for which [CCH+19] directly proved adaptive security of the corresponding Fiat-Shamir NIZK), and thus we view this more as a conceptual contribution providing a possible path towards obtaining efficient Fiat-Shamir NIZKs with adaptive zero knowledge in the future.

1.2 Technical Overview

1.2.1 Adaptive-input SHVZK

Our first compiler exploits so-called instance-dependent trapdoor commitment (IDTC) schemes. Intuitively, this primitive is parameterized by an NP language L and allows a sender to create a commitment `com` (with opening `dec`) to a message m using a statement x as a label. The main idea is that: (i) In case $x \notin L$ is a *false* statement, the commitment satisfies the standard *binding* property. (ii) In case $x \in L$ is a *true* statement, the commitment satisfies the standard *hiding* property and additionally, given a valid witness w for x, one can generate a fake commitment `com` that is distributed like an honest commitment but that can later be opened to any message (the so-called *trapdoorness* property). It is well known that IDTCs for any language L can be constructed in a black-box way given any Sigma protocol for L [DG03, Dam10, Lin15, CPSV16].

We now explain how to compile any delayed-input Sigma protocol Σ for a language L into a delayed-input Sigma protocol Σ'' for L that satisfies adaptive-input SHVZK. The transformation relies on an IDTC Π for the language L_{DH} of Diffie-Hellman (DH) tuples, and on a Sigma protocol Σ' for the complement language $\overline{L}_{DH}$ of non-DH tuples.

- The prover, given only $x \in L$, starts by sampling a random non-DH tuple $T \in \overline{L}_{DH}$, along with the corresponding witness, and then computes a commitment $\mathtt{com}$ (with decommitment $\mathtt{dec}$) to the first round a of Σ using T as label. Next, it computes the first round a' of Σ' and forwards $(\mathtt{com}, a', T)$ to the verifier.
- The verifier sends a random ℓ-bit challenge c to the prover.
- Upon receiving a valid witness w for x, the prover computes the third round z and z' of both Σ and Σ', and forwards them to the verifier along with the opening $(a, \mathtt{dec})$ of commitment $\mathtt{com}$.

The proof of (adaptive-input) SS of Σ'' follows readily from the (adaptive-input) SS of Σ and the binding property of Π. Hence, we here focus on the proof of adaptive-input SHVZK. The simulator proceeds as follows:

- Upon receiving challenge $c \in \{0,1\}^\ell$, the simulator first samples a random DH tuple $T \in L_{DH}$ and generates a fake commitment $\mathtt{com}$ using T and its corresponding witness. Next, it runs the SHVZK simulator of Σ' upon input T and c obtaining (a', z') and returns a simulated first round $a'' = (\mathtt{com}, a', T)$.
- Upon receiving statement $x \in L$, the simulator runs the SHVZK simulator of Σ upon input x and c obtaining (a, z). Hence, it opens the commitment $\mathtt{com}$ to a obtaining decommitment $\mathtt{dec}$, and returns a simulated third round $z'' = (z, z', (a, \mathtt{dec}))$.

In the proof, we first move to a mental experiment with a modified simulator that generates (a, z) using the real prover of Σ; this is possible thanks to the SHVZK property of Σ. Next, we replace $T \in L_{DH}$ with $T \in \overline{L}_{DH}$ and $(\mathtt{com}, \mathtt{dec})$ with an honestly computed commitment to a. The DDH assumption and the trapdoorness property of Π imply that no efficient distinguisher can notice such a change. Finally, we use the SHVZK property of Σ' to compute (a', z') as the real prover of Σ' would do, which yields exactly the same distribution of proofs as generated by our compiler, and thus concludes the proof. Note that, besides running Σ, our transformation requires to run an IDTC in parallel with Σ' (to prove that a tuple is DH). The cost of running the IDTC corresponds to running a Sigma protocol for DH tuples. The cost of running a Sigma protocol that proves that a tuple is DH is of 2 exponentiations for the prover and 4 exponentiations for the verifier. Therefore, our compiler adds an overhead of 4 exponentiations for the prover and 8 exponentiations for the verifier.

1.2.2 Adaptive Security of Fiat-Shamir NIZKs

We start by recalling the notion of trapdoor Sigma protocols in more details. Intuitively, a trapdoor Sigma protocol is a special kind of three-round public-coin proof system in the CRS model[6] with the guarantee that, for every valid CRS ω, every false statement $x \notin L$, and every first round a, there is *at most one* challenge $c := f(\omega, a, x)$ such that, for some z, the transcript (a, c, z) is accepting w.r.t. (ω, x). The function f is called the *bad-challenge function*. Moreover, it is possible to generate an honest-looking CRS ω along with a trapdoor τ which allows to efficiently compute the bad challenge c given the first round a and the statement x.

Let Σ be a trapdoor Sigma protocol for language L, and Π be the non-interactive proof derived from Σ via the Fiat-Shamir transform using a CI hash family $\mathcal{H}$ for all "efficiently searchable" sparse relations. The proof of adaptive security of Π follows closely the approached used in [CCH+19], with a few crucial differences. In particular:

[6] The latter means that, at setup, a CRS ω is generated and distributed to both the prover and the verifier.

- To show adaptive soundness, one first argues that any prover which, given an honestly-computed CRS (ω, k), is able to produce a statement $x \notin L$ and a non-interactive proof $\pi = (a, z)$ such that (a, c, z) is accepting w.r.t. (ω, x) for $c = h_k(a||x)$ with non-negligible probability, must do so even in case the CRS ω of Σ is generated along with the trapdoor τ. The latter, however, contradicts the CI property of the hash family $\mathcal{H}$ w.r.t. the (efficiently searchable) relation $R_{\omega,\tau} := \{(a||x, c) : \ x \notin L \wedge c = f(\tau, \omega, a, x)\}$, which is easily seen to be sparse thanks to the soundness property of Σ. The key observation that allows to prove adaptive security here is the fact that the hash function takes also x as input, which allows the reduction to CI to go through without knowing x in advance.
- The simulator for adaptive zero knowledge picks a random challenge c and then obtains a invoking the adaptive-input SHVZK simulator of Σ. Hence, it samples a fresh CRS ω and a random hash key k from the conditional distribution $h_k(a) = c$, yielding a simulated CRS (ω, k). Finally, upon receiving $x \in L$ from the adversary, it obtains z from the adaptive-input SHVZK simulator and outputs a simulated proof $\pi = (a, z)$. We note that for this to work it is essential that the challenge c is obtained by hashing only the prover's first message a, as otherwise the simulator would not be able to sample k uniformly from the conditional distribution $c = h_k(a||x)$ without being given x in advance.

From Sigma protocols to trapdoor Sigma protocols. Let us now explain our compiler for turning any Sigma protocol Σ into a *trapdoor* Sigma protocol Σ'. The CRS ω' of Σ' consists of the CRS ω of Σ (if any), along with the public key pk of a (committing) public-key encryption (PKE) scheme. For simplicity, let us assume that the challenge space of Σ is $\{0, 1\}$; it is immediate to extend the challenge space arbitrarily using parallel repetition. The prover of Σ' simply obtains a by running the prover of Σ. Hence, it computes both answers z_0 and z_1 corresponding to the two possible challenges $c = 0$ and $c = 1$, and it encrypts z_0 and z_1 under pk obtaining two ciphertexts e_0, e_1. The prover's first message consists of $a' = (a, e_0, e_1)$. Finally, upon receiving a challenge c, the prover's last message z' consists of the response z_c along with the random coins r_c used to obtain e_c. The verifier accepts if and only if (a, c, z_c) is a valid transcript w.r.t. (ω, x), and additionally (z_c, r_c) is consistent with e_c. It is not hard to show that the above transformation preserves both SS and SHVZK of the underlying protocol Σ, so long as the PKE scheme is semantically secure. To prove that Σ' is a trapdoor Sigma protocol it remains to show how to efficiently compute the bad-challenge function. The main idea here is to let the secret key sk corresponding to pk be the trapdoor τ. This way, given a' and x, we can decrypt e_0, e_1 obtaining[7] the responses z_0, z_1. Note that in case both transcripts $(a, 0, z_0)$ and $(a, 1, z_1)$ are accepting w.r.t. (ω, x), the SS property of Σ implies that $x \in L$. On the other hand, if $x \notin L$, there exists at most one challenge c such that (a, c, z_c) is accepting, and we can determine c efficiently by simply running the verifier algorithm upon input both transcripts $(a, 0, z_0)$ and $(a,_1 , z_1)$. Note that, besides running Σ, our transformation requires to encrypt two values using a public-key encryption scheme. Moreover, if we want a trapdoor Sigma protocol with challenge space $\{0, 1\}^\ell$ for some $\ell \in \mathbb{N}$, and consequently better soundness, we need to repeat our protocol in parallel ℓ times. If the cost of running Σ is C_P for the prover and C_V for the verifier, and the cost of computing an encryption is C_E, then the cost of running our protocol for the prover is $(C_P + 2C_E)\ell$ and for the verifier is $(C_V + C_E)\ell$.

Adding adaptive-input SHVZK. Note that Σ' as defined above is inherently not delayed-input, even assuming Σ is delayed-input. This is because the prover needs the witness in order to compute the two possible responses z_0, z_1 already in the first round. Our last compiler overcomes

[7] Due to the committing property of the PKE scheme.

this problem by extending our very first transform (for obtaining delayed-input Sigma protocols with adaptive-input SHVZK) to *trapdoor* Sigma protocols. The main idea is to work in the CRS model and replace the IDTC with an *extractable* IDTC (a new notion that we introduce). Intuitively, the difference between IDTCs and extractable IDTCs is that the latter are defined in the CRS model, and the CRS can be generated together with a trapdoor in such a way that, given a commitment of the extractable IDTC scheme with respect to a false instance, it is possible to extract the committed value (which is unique) using the trapdoor. Moreover, the commitment procedure now outputs a message $\texttt{com}$ and an instance T such that the verifier can check if the first two components of T are consistent with the CRS. As we show, the latter allows to preserve the trapdoorness property of Σ when applying the transformation described before, while at the same time boosting SHVZK to adaptive-input SHVZK. Finally, we give a simple construction of an extractable IDTC Π for the language L_{DH} of DH tuples. This construction is based on the observation that the classical Sigma protocol for DH tuples has a special extractor which, on input the first round $a = (g^r, h^{r'})$ and γ such that $h = g^\gamma$, outputs the only possible challenge c that would make the transcript (a, c, z) accepting w.r.t. a non-DH tuple (for some z). Given a Sigma protocol Σ for L_{DH}, we then show how to obtain an extractable IDTC. The main idea is to set the CRS to $(g, h = g^\gamma)$ and the trapdoor to $\tau = \gamma$. Each commitment $\texttt{com}$ is then equipped with a value $T = (g^\alpha, h^\beta)$ with $\alpha \neq \beta$. Note that in this case (ω, T) corresponds to a non-DH tuple, hence the extractor can be run on $\texttt{com}$, which corresponds to the first round of Σ.

1.3 Applications

Our results directly allow to achieve adaptive security of delayed-input Sigma protocols and Fiat-Shamir NIZKs. Since applications of the latter are well known, below we elaborate on the impact of our results on applications of the former. The delayed-input property directly improves the round complexity of any cryptographic protocol consisting of the following two steps: (1) an NP-statement x and a witness w is defined via an interactive process; and (2) one of the parties involved in the protocol provides a proof that x is a true statement. Indeed, using a delayed-input Sigma protocol that allows proving the validity of x, it is possible to parallelize the above two steps, thus decreasing the round complexity of the overall process. Furthermore, the delayed-input property of FLS has proven to be particularly powerful for providing round-efficient constructions from general assumptions, such as: 4-round (optimal) secure 2PC where only one player gets the output (5 rounds when both players get the output) [KO04], 4-round 2PC in the simultaneous message exchange model where both parties get the output [COSV17b], 4-round MPC for any functionality [CCG+19, BGJ+18, ACJ17, BHP17], 3-round non-malleable commitments [COSV16, GR19] and 4-round non-malleable commitments [COSV17a, GRRV14]. In many cryptographic applications, one party needs to prove an OR statement of the form *"either x is true or I know a trapdoor"*, where neither x nor the trapdoor might be known at the beginning of the protocol. Our adaptive-input SHVZK Sigma protocols can be used to prove exactly this kind of statements, as we can combine adaptive-input (and non-adaptive-input) SHVZK Sigma protocols using the well-known OR composition technique of [CDS94], which yields an adaptive witness-indistinguishable (WI) Sigma protocol (*i.e.*, a Sigma protocol that retains the WI property even when the statement is adaptively chosen after the first round). The notion of adaptive WI was formalized in [CPS+16], where the authors proposed a general compiler to obtain this property. The advantage of our approach is that we obtain a more efficient compiler. Indeed, the compiler of [CPS+16] requires computing at least one additional commitment for each statement that composes the OR theorem.

2 Preliminaries

We start the section by introducing our notation. For a string x, we denote its length by $|x|$; if S is a set, $|S|$ represents the number of elements in S. When x is chosen randomly in S, we write $x \leftarrow\!\!\$\ S$. When $\mathcal{A}$ is a randomized algorithm, we write $y \leftarrow\!\!\$\ \mathcal{A}(x)$ to denote a run of $\mathcal{A}$ on input x (and implicit random coins r) and output y; the value y is a random variable, and $\mathcal{A}(x;r)$ denotes a run of $\mathcal{A}$ on input x and randomness r. An algorithm $\mathcal{A}$ is *probabilistic polynomial-time* (PPT) if $\mathcal{A}$ is randomized and for any input $x, r \in \{0,1\}^*$ the computation of $\mathcal{A}(x;r)$ terminates in a polynomial number of steps (in the size of the input). A *polynomial-time* relation R is a relation for which membership of (x,w) w.r.t. R can be decided in time polynomial in $|x|$. If $(x,w) \in R$ then we say that w is a *witness* for *instance* x. A polynomial-time relation R is naturally associated with the NP language L_R defined as $L_R = \{x : \exists w \text{ s.t. } (x,w) \in R\}$. (When R is clear from the context, we simply write L.) Similarly, an NP language is naturally associated with a polynomial-time relation. We denote by $\hat{L}_R$ the language such that $L_R \subseteq \hat{L}_R$ and membership in $\hat{L}_R$ may be tested in polynomial time.

2.1 Standard Definitions

Negligible functions. We denote with $\lambda \in \mathbb{N}$ the security parameter. A function p is a polynomial if $p(\lambda) \in O(\lambda^c)$ for some constant $c > 0$. A function $\nu : \mathbb{N} \to [0,1]$ is negligible in the security parameter (or simply negligible) if it vanishes faster than the inverse of any polynomial in λ, *i.e.* $\nu(\lambda) \in O(1/p(\lambda))$ for all positive polynomials $p(\lambda)$.

Throughout the paper, we often implicitly assume that the security parameter is given as input (in unary) to all algorithms.

Random variables. For a random variable $\mathbf{X}$, we write $\mathbb{P}[\mathbf{X} = x]$ for the probability that $\mathbf{X}$ takes on a particular value $x \in X$ (with X being the set where $\mathbf{X}$ is defined). The statistical distance between two random variables $\mathbf{X}$ and $\mathbf{X}'$ with the same domain X is defined as $\Delta(\mathbf{X};\mathbf{X}') = \frac{1}{2}\sum_{x\in X} |\mathbb{P}[\mathbf{X} = x] - \mathbb{P}[\mathbf{X}' = x]|$.

Given two ensembles $\mathbf{X} = \{\mathbf{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathbf{Y} = \{\mathbf{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, we write $\mathbf{X} \equiv \mathbf{Y}$ to denote that they are identically distributed, $\mathbf{X} \approx_s \mathbf{Y}$ to denote that they are statistically close, *i.e.* $\Delta(\mathbf{X}_\lambda;\mathbf{X}'_\lambda) \leq \nu(\lambda)$ for a negligible function $\nu(\lambda)$, and $\mathbf{X} \approx_c \mathbf{Y}$ to denote that they are computationally indistinguishable, *i.e.* for all PPT distinguishers $\mathcal{D}$ it holds that $|\mathbb{P}[\mathcal{D}(\mathbf{X}_\lambda) = 1] - \mathbb{P}[\mathcal{D}(\mathbf{Y}_\lambda) = 1]| \leq \nu(\lambda)$ for a negligible function $\nu(\lambda)$.

2.2 Public-Key Encryption

Definition 1 (Public-key encryption scheme)**.** A triple of PPT algorithms $\Pi := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is called a *public-key encryption* (PKE) scheme if it satisfies the following properties.

Correctness: For every $\lambda \in \mathbb{N}$, every $(sk, pk) \in \mathsf{KGen}(1^\lambda)$, and every $m \in \{0,1\}^\lambda$, we have that $\mathbb{P}[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m] = 1$.

Semantic security: Let $\mathcal{A}$ be PPT adversary. Consider the following experiments:

Experiment $\mathbf{Exp}^{\mathsf{ind\text{-}0}}_{\Pi,\mathcal{A}}(\lambda)$:	Experiment $\mathbf{Exp}^{\mathsf{ind\text{-}1}}_{\Pi,\mathcal{A}}(\lambda)$:
$(pk, sk) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$	$(pk, sk) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$
$(m_0, m_1, st) \leftarrow\!\!\$\ \mathcal{A}(pk)$	$(m_0, m_1, st) \leftarrow\!\!\$\ \mathcal{A}(pk)$
$b' \leftarrow\!\!\$\ \mathcal{A}(st, \mathsf{Enc}(pk, m_0))$	$b' \leftarrow\!\!\$\ \mathcal{A}(st, \mathsf{Enc}(pk, m_1))$
return b'	return b'

We say that Π is secure if for every PPT $\mathcal{A}$ there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that:

$$\left| \mathbb{P}\left[\mathbf{Exp}^{\mathsf{ind\text{-}1}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] - \mathbb{P}\left[\mathbf{Exp}^{\mathsf{ind\text{-}0}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] \right| \leq \nu(\lambda).$$

2.3 Sigma Protocols

Let L be an NP language, with corresponding relation R. A Sigma protocol $\Sigma = (\mathcal{P}, \mathcal{V})$ for R is a 3-round public-coin protocol. In particular, an execution of Σ proceeds as follows:

- The prover $\mathcal{P}$ computes the first message using as input the instance to be proved $x \in L$ with the corresponding witness w, and outputs the first message a with an auxiliary information st; we denote this action with $(a, st) \leftarrow\!\!\$\ \mathcal{P}(x, w)$.
- The verifier $\mathcal{V}$, upon receiving a, sends a random string $c \leftarrow\!\!\$\ \{0,1\}^\ell$ with $\ell \in \mathbb{N}$.
- The prover $\mathcal{P}$, upon input c and st, computes and sends z to $\mathcal{V}$; we denote this action with $z \leftarrow\!\!\$\ \mathcal{P}(st, c)$.
- The verifier $\mathcal{V}$, upon input (x, a, c, z), outputs 1 to accept and 0 to reject; we denote this action with $\mathcal{V}(x, a, c, z) = d$ where $d \in \{0,1\}$ denotes whether $\mathcal{V}$ accepts or not.

Definition 2 (Sigma protocol [CDS94])**.** A 3-move protocol Σ with challenge length $\ell \in \mathbb{N}$ is a *Sigma protocol* for a relation R if it enjoys the following properties:

- **Completeness.** If $(x, w) \in R$, then all honest 3-move transcripts for (x, w) are accepting.
- **Special soundness.** There exists an efficient algorithm $\mathcal{K}$ that, on input two accepting transcripts (a, c, z) and (a, c', z') for x with $c' \neq c$ (we refer to such two accepting transcripts as a *collision*) outputs a witness w such that $(x, w) \in R$.
- **Special honest-verifier zero knowledge (SHVZK).** There exists a PPT simulator algorithm $\mathcal{S}$ that takes as input security parameter 1^λ, $x \in L$ and $c \in \{0,1\}^\ell$, and outputs an accepting transcript for x where c is the challenge (we denote this action with $(a, z) \leftarrow\!\!\$\ \mathcal{S}(x, c)$). 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 3-move honest transcript obtained when $\mathcal{V}$ sends c as challenge and $\mathcal{P}$ runs on common input x and any private input w such that $(x, w) \in R$.

2.4 The DDH Assumption

Let $\mathcal{G}$ be a cyclic group with generator g, and let A, B and X be elements of $\mathcal{G}$. We say that (g, A, B, X) is a *Diffie-Hellman tuple* (a *DH tuple*, in short) if $A = g^\alpha, B = g^\beta$ for some integers $0 \leq \alpha, \beta \leq |\mathcal{G}| - 1$ and $X = g^{\alpha\beta}$. If this is not the case, the tuple is called *non-DH*. To verify that a tuple is DH, it is sufficient to have the discrete log α of A to the base g, and then to check that $X = B^\alpha$. We thus define the polynomial-time relation $R_{DH} = \{((g, A, B, X), \alpha) : A = g^\alpha \text{ and } X = B^\alpha\}$ of all DH tuples.

The *Decisional Diffie-Hellman* assumption (the DDH assumption) posits the hardness of distinguishing a randomly selected DH tuple from a randomly selected non-DH tuple with respect to a *group generator* algorithm. For sake of concreteness, we consider the specific group generator GG that, on input 1^λ, randomly selects a λ-bit prime p such that $q = (p-1)/2$ is also prime and outputs the (description of the) order q group $\mathcal{G}$ of the quadratic residues modulo p along with a random generator g of $\mathcal{G}$.

Assumption 1 (DDH Assumption)**.** *For every PPT algorithm $\mathcal{A}$ there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ s.t.*

$$\Big|\mathbb{P}\left[\mathcal{A}((\mathcal{G},q,g),g^{\alpha},g^{\beta},g^{\gamma})=1 : (\mathcal{G},q,g)\leftarrow_{\$}\mathsf{GG}(1^{\lambda});\alpha,\beta,\gamma\leftarrow_{\$}\mathbb{Z}_q\right]-$$
$$\mathbb{P}\left[\mathcal{A}((\mathcal{G},q,g),g^{\alpha},g^{\beta},g^{\alpha\beta})=1 : (\mathcal{G},q,g)\leftarrow_{\$}\mathsf{GG}(1^{\lambda});\alpha,\beta,\gamma\leftarrow_{\$}\mathbb{Z}_q\right]\Big|\leq\nu(\lambda).$$

1-non-DDH. We define polynomial-time relation R_{1nDH} as $R_{1nDH} = \{((g,A,B,X),\alpha) : A = g^{\alpha} \text{ and } X = g \cdot B^{\alpha}\}$. Thus a *1-non-DH* tuple is a tuple $T = (g,A,B,X)$ such that $A = g^{\alpha}$, $B = g^{\beta}$ and $X = g \cdot B^{\alpha} = g^{\alpha\cdot\beta+1}$. Under the DDH assumption random 1-non-DH tuples are indistinguishable from random non-DH tuple. Indeed, let $T = (g,A,B,X)$ be any tuple and consider tuple $T' = (g,A,B,g\cdot X)$. Then we have that if T is a randomly selected DH tuple, then T' is a randomly selected 1-non-DH tuple; on the other hand, if T is a randomly selected non-DH tuple then T' is statistically close to a randomly selected non-DH tuple. By transitivity, we have that, under the DDH assumption, randomly selected 1-non-DH tuples are indistinguishable from randomly selected DH tuples We can thus state the following lemma (first proved as Lemma 3.1 in [HKR+14]).

Lemma 1 ([HKR+14])**.** *Under the DDH assumption, for every PPT algorithm $\mathcal{A}$ there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ s.t.*

$$\Big|\mathbb{P}\left[\mathcal{A}((\mathcal{G},q,g),g^{\alpha},g^{\beta},g^{\alpha\beta+1})=1 : (\mathcal{G},q,g)\leftarrow_{\$}\mathsf{GG}(1^{\lambda});\alpha,\beta\leftarrow_{\$}\mathbb{Z}_q\right]-$$
$$\mathbb{P}\left[\mathcal{A}((\mathcal{G},q,g),g^{\alpha},g^{\beta},g^{\alpha\beta})=1 : (\mathcal{G},q,g)\leftarrow_{\$}\mathsf{GG}(1^{\lambda});\alpha,\beta\leftarrow_{\$}\mathbb{Z}_q\right]\Big|\leq\nu(\lambda).$$

As showed in [CPS+16], a Sigma protocol Σ_{1nDH} for the relation R_{1nDH} can be constructed based on the Sigma protocol Σ_{DH} of [CDS94] to prove that a given tuple is DH. The compiler proposed in [CPS+16] is almost as efficient as Σ_{DH} and works as follows. On input tuples (g,A,B,X), the prover and the verifier construct tuples (g,A,B,Y) by setting $Y = X/g$. Then, they simply run Sigma protocol Σ_{DH} upon input the theorem (g,A,B,Y).

Theorem 1 ([CPS+16])**.** *Protocol Σ_{1nDH} is a Sigma protocol for the relation R_{1nDH}.*

2.5 Instance-dependent Trapdoor Commitment

An *instance-dependent trapdoor commitment* scheme for polynomial-time relation R with message space M is a quadruple of PPT algorithms $(\mathsf{Com},\mathsf{Dec},\mathsf{Fake}_1,\mathsf{Fake}_2)$ specified as follows:

- Com is the randomized *commitment* algorithm that takes as input an instance $x \in \hat{L}$ and a message $m \in M$, and outputs *commitment* $\mathtt{com}$ and *decommitment* $\mathtt{dec}$;
- Dec is the *verification* algorithm that takes as input $x \in \hat{L}$, $\mathtt{com},\mathtt{dec}$ and $m \in M$, and decides whether m is the decommitment of $\mathtt{com}$;
- Fake_1 takes as input $(x,w) \in R$ and outputs *commitment* $\mathtt{com}$, and *equivocation information* $\mathtt{rand}$;
- Fake_2 takes as input $(x,w) \in R$, message $m \in M$, and $(\mathtt{com},\mathtt{rand})$, and outputs $\mathtt{dec}$;

Definition 3 (Instance-dependent trapdoor commitment scheme)**.** Let R be a polynomial-time relation. We call $\Pi = (\mathsf{Com},\mathsf{Dec},\mathsf{Fake}_1,\mathsf{Fake}_2)$ an *instance-dependent trapdoor commitment* scheme (an *IDTC*, in short) for R if it enjoys the following properties:

- **Correctness.** For all $x \in \hat{L}$, and all $m \in M$, it holds that

$$\mathbb{P}\left[\mathsf{Dec}(x,\mathtt{com},\mathtt{dec},m)=1 :\ (\mathtt{com},\mathtt{dec})\leftarrow_{\$}\mathsf{Com}(x,m)\right]=1.$$

- **Binding.** For all $x \notin L$, and for every commitment $\mathtt{com}$, there exists at most one message $m \in M$ for which there is a valid decommitment $\mathtt{dec}$ (*i.e.* $\mathsf{Dec}(x, \mathtt{com}, \mathtt{dec}, m) = 1$).
- **Hiding.** For every $x \in L$, and every $m_0, m_1 \in M$, the ensembles $\{\mathtt{com} : (\mathtt{com}, \mathtt{dec}) \leftarrow\!\!\$\ \mathsf{Com}(1^\lambda, x, m_0)\}_{\lambda\in\mathbb{N}}$ and $\{\mathtt{com} : (\mathtt{com}, \mathtt{dec}) \leftarrow\!\!\$\ \mathsf{Com}(1^\lambda, x, m_1)\}_{\lambda\in\mathbb{N}}$ are identically distributed.
- **Trapdoorness.** For all $(x, w) \in R$ and $m \in M$ the following two distributions coincide:

$$\{(\mathtt{com}, \mathtt{dec}) :\ (\mathtt{com}, \mathtt{rand}) \leftarrow\!\!\$\ \mathsf{Fake}_1(x, w); \mathtt{dec} \leftarrow\!\!\$\ \mathsf{Fake}_2(x, w, m, \mathtt{com}, \mathtt{rand})\}$$
$$\{(\mathtt{com}, \mathtt{dec}) :\ (\mathtt{com}, \mathtt{dec}) \leftarrow \mathsf{Com}(x, m)\}\,.$$

An IDTC can be easily constructed from any Sigma protocol as shown in [Dam10, HL10, DN02, CPS+16]. For the sake of completeness we now recall the construction of [CPS+16].

IDTCs from Sigma protocols. Let $\Sigma = (\mathcal{P}, \mathcal{V})$ be a Sigma protocol for the polynomial-time relation R, with challenge length ℓ, and SHVZK simulator $\mathcal{S}$. We define an IDTC $\Pi = (\mathsf{Com}^\Sigma, \mathsf{Dec}^\Sigma, \mathsf{Fake}_1^\Sigma, \mathsf{Fake}_2^\Sigma)$ with message space $M = \{0,1\}^\ell$ as follows.

- Com^Σ takes as input x and $m \in \{0,1\}^\ell$ and computes $(\mathtt{com}, \mathtt{dec})$ by running $\mathcal{S}$ on input (x, m).
- Dec^Σ takes as input $x, \mathtt{com}, \mathtt{dec}, m$, runs $\mathcal{V}$ on instance x and transcript $(\mathtt{com}, m, \mathtt{dec})$ and outputs $\mathcal{V}$'s output.
- Fake_1^Σ takes as input $(x, w) \in R$, samples a random string ρ and runs $\mathcal{P}$ on input (x, w) and randomness ρ to get the 1st message a of Σ. Then, it sets $\mathtt{com} = a$ and $\mathtt{rand} = \rho$ and outputs $(\mathtt{com}, \mathtt{rand})$.
- Fake_2^Σ takes as input $(x, w) \in R$, message $m \in \{0,1\}^\ell$, and commitment $\mathtt{com}$ and equivocation information $\mathtt{rand}$ and runs $\mathcal{P}$ to get 3rd message z of Σ corresponding to first message $a = \mathtt{com}$, randomness $\rho = \mathtt{rand}$ and challenge $c = m$. Then, it sets $\mathtt{dec} = z$ and outputs $\mathtt{dec}$.

Theorem 2 ([CPS+16])**.** *Assuming Σ is a Sigma protocol for the relation R, the above construction Π^Σ defines an IDTC for R.*

Proof sketch. The security proof relies only on the properties of Σ. Correctness follows from the completeness of Σ. Binding follows from the special soundness of Σ. Hiding and Trapdoorness follow from the SHVZK and the completeness of Σ. □

2.6 Correlation Intractable Hash Families

Here, we recall the definition of correlation-intractable hash families for all sparse relations.

Definition 4 (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) $\mathcal{H}.\mathsf{Hash}(k, x)$ computes the function $h_k(x)$.

Definition 5 (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 $\mathcal{A} := \{\mathcal{A}_\lambda\}$:

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

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

Correlation intractability is a useful and versatile property of random oracles that we would like to guarantee in the standard model. However, even a random oracle is only R-correlation intractable for so-called *sparse* relations.

Definition 6 (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 achieving 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 7 (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 8 (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 $poly(n)$.

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

Definition 9 (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 $\mathbb{P}\left[h_k(x) = y : k \leftarrow\!\!\$\ \mathcal{H}.\mathsf{Gen}(1^\lambda)\right] = 2^{-m(\lambda)}$.

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

2.7 Non-Interactive Argument Systems

Finally, we recall the definitions related to non-interactive argument systems.

Definition 10 (NIZK argument systems)**.** A *non-interactive zero-knowledge argument system (NIZK)* for an NP-language L consists of three PPT machines $\Pi := (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$, that have the following properties:

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

$$\mathbb{P}\left[\mathcal{V}(\omega, x, \mathcal{P}(\omega, x, w)) = 1 : \omega \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda, 1^{|x|})\right] = 1.$$

- **Soundness.** For all PPT provers $\mathcal{P}^*$, there exists a negligible function $\nu : \mathbb{N} \to [0,1]$, such that for all $\lambda \in \mathbb{N}$ and for all $x \notin L$:

$$\mathbb{P}\left[\mathcal{V}(\omega, x, \pi) = 1 : \omega \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda, 1^{|x|}); \pi \leftarrow\!\!\$\ \mathcal{P}^*(\omega)\right] \leq \nu(\lambda).$$

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

A NIZK argument system can also satisfy various stronger properties. We list them below.

- **Adaptive zero knowledge.** For all PPT verifiers $\mathcal{V}^*$ there exists a PPT simulator $\mathcal{S} := (\mathcal{S}_0, \mathcal{S}_1)$ such that the following distribution ensembles are computationally indistinguishable:

$$\{(\omega, \pi) : \omega \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda, 1^{|x|}); (x, w) \leftarrow\!\!\$\ \mathcal{V}^*(\omega); \pi \leftarrow\!\!\$\ \mathcal{P}(\omega, x, w); (x, w) \in R\}_{\lambda \in \mathbb{N}}$$
$$\{(\omega, \pi) : (\omega, \tau) \leftarrow\!\!\$\ \mathcal{S}_0(1^\lambda, 1^{|x|}); (x, w) \leftarrow\!\!\$\ \mathcal{V}^*(\omega); \pi \leftarrow\!\!\$\ \mathcal{S}_1(\omega, \tau, x); (x, w) \in R\}_{\lambda \in \mathbb{N}}$$

- **Adaptive soundness.** For all PPT prover $\mathcal{P}^*$, there exists a negligible function $\nu : \mathbb{N} \to [0, 1]$, such that for all $\lambda \in \mathbb{N}$:

$$\mathbb{P}\left[\mathcal{V}(\omega, x, \pi) = 1 : \omega \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda, 1^{|x|}); (x, \pi) \leftarrow\!\!\$\ \mathcal{P}^*(\omega); x \notin L\right] \leq \nu(\lambda).$$

3 A Compiler for Adaptive-input HVZK

3.1 Delayed-input Sigma Protocols

In this section, we recall the notion of a *delayed-input* three-round public-coin protocols and give security notions both for the prover and the verifier.

Definition 11 (Delayed-input protocols [CPS+16])**.** A *delayed-input* three-move protocol for polynomial-time relation R is a three-move protocol $(\mathcal{P}, \mathcal{V})$ in which the first message of $\mathcal{P}$ can be computed on input the length n of the common theorem in unary notation.[8]

Since in a delayed-input protocol the input might not be fixed at the onset of the protocol, it could be adversarially chosen by the prover or by the verifier based on the exchanged messages. For example, the special soundness of a Sigma protocol guarantees extraction from collisions only if the transcripts are accepting for the same input. However when the statement to be proved is chosen adaptively by the prover depending on the challenge received, special soundness does not guarantee extraction. We thus introduce a stronger notion that we call *adaptive-input special soundness*. Roughly speaking, we require that it is possible to extract witnesses from a collision even if the two accepting transcripts are for different statements.

Definition 12 (Adaptive-input special soundness)**.** A delayed-input 3-round protocol $\Sigma = (\mathcal{P}, \mathcal{V})$ for relation R enjoys *adaptive-input special soundness* if there exists a polynomial-time algorithm $\mathcal{K}$ such that, for any $x_1, x_2 \in L$, and for any pair of accepting transcripts (a, c_1, z_1) for input x_1 and (a, c_2, z_2) for input x_2 with $c_1 \neq c_2$, outputs witnesses w_1 and w_2 such that $(x_1, w_1) \in R$ and $(x_2, w_2) \in R$.

Similarly, the definition below captures an adaptive flavour of SHVZK where the distinguisher can choose the theorem being proven adaptively based on the honest prover's first message and on the verifier's challenge.

Definition 13 (Adaptive-input SHVZK)**.** A delayed-input 3-round protocol $\Sigma = (\mathcal{P}, \mathcal{V})$ for relation R satisfies *adaptive-input special honest-verifier zero-knowledge (adaptive-input SHVZK)* if there exists a PPT simulator algorithm $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ such that for all PPT adversaries $\mathcal{A}$ and for all challenges $c \in \{0, 1\}^\ell$ there is a negligible function $\nu : \mathbb{N} \to [0, 1]$ for which $\left|\mathbb{P}\left[b' = b\right] - \frac{1}{2}\right| \leq \nu(\lambda)$ in the following game:

1. The challenger sends (a, c) to $\mathcal{A}$, where the value a is either computed using $(a, st) \leftarrow\!\!\$\ \mathcal{P}(1^\lambda, 1^n)$ (in case $b = 0$) or $(a, st) \leftarrow\!\!\$\ \mathcal{S}_0(1^\lambda, 1^n, c)$ (in case $b = 1$).

[8] For simplicity, in what follows, we sometimes drop input 1^n when describing the prover of a delayed-input Sigma protocol.

2. The adversary $\mathcal{A}$ sends a pair (x, w) to the challenger, where $|x| = n$. Hence:
   - If $(x, w) \in R$, the challenger sends z to $\mathcal{A}$, where the value z is either computed using $z \leftarrow\!\!\$\ \mathcal{P}(x, w, st, c)$ (in case $b = 0$) or $z \leftarrow\!\!\$\ \mathcal{S}_1(x, st)$ (in case $b = 1$);
   - Else, the challenger sends $z = \perp$ to $\mathcal{A}$.
3. The adversary $\mathcal{A}$ outputs a bit b'.

3.2 The Transformation

It turns out that the celebrated Sigma protocol by Lapidot and Shamir [LS91] is already delayed-input, and moreover it satisfies both adaptive-input special soundness[9] and adaptive-input SHVZK. While this protocol works for any NP relation, it is very inefficient as it requires generic NP reductions. Hence, it is a natural question whether there are efficient Sigma protocols that are delayed-input and satisfy both adaptive-input special soundness and SHVZK.

A partial answer to this question was given in [CPS+16], which shows how to transform a large class of delayed-input Sigma protocols into ones with adaptive-input special soundness. In this section, we give yet another transform that allows to turn *any* delayed-input Sigma protocol into one satisfying adaptive-input SHVZK. Moreover, assuming the initial Sigma protocol already satisfies adaptive-input special soundness, our transformation preserves this property.

Let Σ be a delayed-input Sigma protocol for a polynomial-time relation R. We construct a Sigma protocol Σ'' for R based on the following additional building blocks: (i) An IDTC $\Pi = (\mathsf{Com}, \mathsf{Dec}, \mathsf{Fake}_1, \mathsf{Fake}_2)$ for the relation R_{DH} (see §2.4); and (ii) A Sigma protocol $\Sigma' = (\mathcal{P}', \mathcal{V}')$ for the relation R_{1nDH} (see §2.4). The formal construction is showed in Fig. 1.

Intuitively, the prover starts by computing the first round a of the Sigma protocol Σ. Hence, it commits to message a using the IDTC with a random 1-non-DH tuple $T \in L_{1nDH}$ as instance (*i.e.*, $A = g^\alpha$, $B = g^\beta$ and $C = g \cdot B^\alpha = g^{\alpha \cdot \beta + 1}$ for random $\alpha, \beta \in \mathbb{Z}_q$)), obtaining a commitment $\mathtt{com}$ and decommitment $\mathtt{dec}$. Next, the prover computes the first round a' of the Sigma protocol Σ' for showing that T is indeed a 1-non-DH tuple, and sends $(\mathtt{com}, a', T)$ to the verifier, which replies with a random challenge $c \in \{0,1\}^\ell$. Finally, the prover completes the transcripts of both Σ and Σ' using c as challenge, obtaining values z, z' that are forwarded to the verifier together with the decommitment information $(\mathtt{dec}, a)$ corresponding to commitment $\mathtt{com}$. We now state and prove our formal theorem.

Theorem 3. *Let R be a polynomial-time relation. Assume that Σ is a delayed-input Sigma protocol for R with adaptive-input special soundness and standard SHVZK, that Π is an IDTC for R_{DH}, and that Σ' is a Sigma protocol for the relation R_{1nDH}. Then, the Sigma protocol Σ'' of Fig. 1 is a delayed-input Sigma protocol for R satisfying both adaptive-input special soundness and adaptive-input SHVZK under the DDH assumption.*

Proof. The completeness of Σ'' follows readily by completeness of Σ and Σ', and by correctness of Π. Below, we thus focus on the proofs of adaptive-input special soundness and adaptive-input SHVZK. Intuitively, our protocol works because the first message of Π is committed using an IDTC that is perfectly binding (since the instance used for the IDTC is non-DH). To enforce a cheating prover to not use a DH tuple (instead of a 1-non-DH one) to compute the commitment, $\mathcal{P}''$ proves that T is a 1-non-DH tuple using the Sigma protocol Σ'. We recall that Σ' can be easily constructed following the approach described in §2.4.

It is important to observe that even though the Sigma protocol Π has a perfect SHVZK simulator, the Sigma protocol Σ'' that our compiler outputs is only (adaptive-input) *computational* SHVZK, and in fact Theorem 3 relies on the hardness of the DDH assumption.

[9] Strictly speaking, [LS91] only achieves a weaker flavor of adaptive-input special soundness that allows to extract the witness for only one of the two theorems.

Common input: Security parameter λ, instance $x \in L$ (available only in the third round).
Prover's private input: Witness w s.t. $(x, w) \in R$ (available only in the third round).
$\mathcal{P}''(1^\lambda)$**:** The first prover's message is generated as follows:

- Pick a random 1-non-DH tuple $T = (g, A, B, C)$ with corresponding witness $\alpha \in \mathbb{Z}_q$.
- Run $\mathcal{P}$ on input $(1^\lambda, 1^n)$ thus obtaining (a, st).
- Run Com on input T and a, thus obtaining $(\mathtt{com}, \mathtt{dec})$.
- Run $\mathcal{P}'$ on input (T, α) thus obtaining (a', st').
- Send $a'' = (\mathtt{com}, a', T)$ to $\mathcal{V}''$ and set $st'' = (st, st', \mathtt{dec}, a)$.

$\mathcal{V}''(1^\lambda)$**:** Pick $c \leftarrow\!\!\$\ \{0,1\}^\ell$ and send it to $\mathcal{P}''$.
$\mathcal{P}''(x, w, st'', c)$**:** The last prover's message is generated as follows:

- Run $\mathcal{P}$ on input (x, w, st, c) thus obtaining z
- Run $\mathcal{P}'$ on input (c, st') thus obtaining z'.
- Send $z'' = (z, z', (\mathtt{dec}, a))$ to $\mathcal{V}'$.

Verifier's verdict: Upon input a transcript (a'', c, z'') such that $a'' = (\mathtt{com}, a', T)$ and $z'' = (z, z', (\mathtt{dec}, a))$ the verifier accepts if and only if the following checks go through: (i) $\mathcal{V}(x, a, c, z) = 1$; (ii) $\mathcal{V}'(T, a', c, z') = 1$; (iii) $\mathsf{Dec}(T, \mathtt{com}, \mathtt{dec}, a) = 1$.

Figure 1: Construction of a delayed-input Sigma protocol with adaptive-input SHVZK.

Proof of adaptive-input special soundness. We start by showing that Σ'' satisfies adaptive-input special soundness.

Lemma 2. *Assuming that* Π *satisfies binding, and that* Σ *(resp.* Σ'*) satisfies special soundness (resp. adaptive-input special soundness), then the delayed-input protocol* Σ'' *of Fig. 1 satisfies adaptive-input special soundness.*

Proof. Fix any $x_1, x_2 \in L$. Let (a'', c_1, z_1'') and (a'', c_2, z_2'') be, respectively, two accepting transcripts for x_1 and x_2, with $c_1 \neq c_2$, and let us write $a'' = (\mathtt{com}, a', T)$, $z_1'' = (z_1, z_1', (\mathtt{dec}_1, a_1))$ and $z_2'' = (z_2, z_2', (\mathtt{dec}_2, a_2))$ as described in Fig. 1.

The fact that both (a'', c_1, z_1'') and (a'', c_2, z_2'') are accepting implies that both (a', c_1, z_1') and (a', c_2, z_2') are accepting w.r.t. statement T under Σ'. Hence, by special soundness of Σ', there exists an efficient algorithm that given the above transcripts returns a valid witness α such that $(T, \alpha) \in R_{1nDH}$.

Since the above is equivalent to saying that $T \notin L_{DH}$, the perfect binding property of Π now implies that $a_1 = a_2 = a$. The latter, in turn, implies that (a, c_1, z_1) and (a, c_2, z_2) are accepting transcripts w.r.t. statements x_1 and x_2 under Σ, with $c_1 \neq c_2$, and thus by adaptive-input special soundness of Σ there exists an efficient algorithm that given the above transcripts returns witnesses w_1, w_2 such that $(x_1, w_1) \in R$ and $(x_2, w_2) \in R$. The statement follows. □

Proof of adaptive-input SHVZK. Next, we show that Σ'' satisfies adaptive-input SHVZK.

Lemma 3. *Assuming that* Π *satisfies trapdoorness, and that* Σ *and* Σ' *satisfy SHVZK, then the protocol* Σ'' *of Fig. 1 satisfies adaptive-input SHVZK under the DDH assumption.*

Proof. We start by describing the adaptive-input SHVZK simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$, in Fig. 2. Intuitively, in the first phase, simulator $\mathcal{S}_0$ generates a random DH tuple $T \in L_{DH}$ (*i.e.*, $A = g^\alpha$, $B = g^\beta$ and $C = B^\alpha = g^{\alpha \cdot \beta}$ for random $\alpha, \beta \in \mathbb{Z}_q$) along with the corresponding witness α, and lets $\mathtt{com}$ be an equivocal commitment w.r.t. (T, α), with equivocation information $\mathtt{rand}$. Furthermore, $\mathcal{S}_0$ simulates a transcript (a', c, z') using the SHVZK simulator $\mathcal{S}'$ of Σ' upon input statement T and challenge c. This yields a simulated first message $a'' = (\mathtt{com}, a', T)$. In the second phase, when the statement $x \in L$ is available, simulator $\mathcal{S}_1$ uses the SHVZK simulator $\mathcal{S}$ of Σ upon input statement x and challenge c to obtain a simulated transcript (a, c, z). Finally, $\mathcal{S}_1$ opens the commitment to a using $\mathtt{rand}$, which yields a simulated response $(z, z', (\mathtt{dec}, a))$.

Let $\mathcal{S}$ and $\mathcal{S}'$ be, respectively, the SHVZK simulators of Σ and Σ'. The simulator $\mathcal{S}'' = (\mathcal{S}''_0, \mathcal{S}''_1)$ for Σ'' proceeds as follows. Recall that $\mathcal{S}''_0$ only gets as input $(1^\lambda, c)$, whereas $\mathcal{S}''_1$ additionally receives the statement $x \in L$.

$\mathcal{S}''_0(1^\lambda, c)$**:** Upon input security parameter 1^λ and the challenge c:

- Pick a random DH tuple $T = (g, A, B, C)$ with corresponding witness $\alpha \in \mathbb{Z}_q$.
- Run $(\mathtt{com}, \mathtt{rand}) \leftarrow\!\!\$\ \mathsf{Fake}_1(T, \alpha)$.
- Run $(a', z') \leftarrow\!\!\$\ \mathcal{S}'(T, c)$.
- Output $a'' = (\mathtt{com}, a', T)$ and $st'' = (T, \alpha, \mathtt{com}, \mathtt{rand}, c, z')$.

$\mathcal{S}''_1(x, st'')$**:** Upon input statement $x \in L$ and state $st'' = (T, \alpha, \mathtt{com}, \mathtt{rand}, c, z')$:

- Run $(a, z) \leftarrow\!\!\$\ \mathcal{S}(x, c)$.
- Run $\mathtt{dec} \leftarrow\!\!\$\ \mathsf{Fake}_2((T, \alpha), a, (\mathtt{com}, \mathtt{rand}))$.
- Output $z'' = (z, z', (\mathtt{dec}, a))$.

Figure 2: The adaptive-input SHVZK simulator of Σ''.

Let us denote with $\mathbf{G}(\lambda, c, 0)$ (resp. $\mathbf{G}(\lambda, c, 1)$) the game of Definition 13 when the challenger uses challenge $c \in \{0,1\}^\ell$ and hidden bit $b = 0$ (resp. $b = 1$). Recall that in $\mathbf{G}(\lambda, c, 0)$ the attacker $\mathcal{A}$ receives a real transcript (a'', c, z'') as computed by the prover $\mathcal{P}''$, whereas in $\mathbf{G}(\lambda, c, 1)$, the attacker $\mathcal{A}$ receives a simulated transcript (a'', c, z'') as computed by the above simulator $\mathcal{S}''$; in both games, the attacker chooses $(x, w) \in R$ adaptively, after obtaining the messages (a'', c). Our goal is to show that $\{\mathbf{G}(\lambda, c, 0)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{G}(\lambda, c, 1)\}_{\lambda\in\mathbb{N}}$, for all $c \in \{0,1\}^\ell$. To this end, we introduce a sequence of hybrids as informally described below and depicted in Fig. 6 on page 34.[10]

Hybrid $\mathbf{H}_1(\lambda, c)$: This experiment is identical to $\mathbf{G}(\lambda, c, 1)$ except that the transcript (a, c, z) is computed using the prover and verifier of Σ instead of running the SHVZK simulator $\mathcal{S}$. Note that the latter is indeed possible as Σ is delayed-input.

Hybrid $\mathbf{H}_2(\lambda, c)$: This experiment is identical to the previous hybrid except that the challenger now picks T from L_{1nDH} (along with the corresponding witness α), and lets $\mathtt{com}$ be a binding commitment to a w.r.t. T (with corresponding decommitment $\mathtt{dec}$).

The claims below conclude the proof of the lemma.

Claim 1. *For all $c \in \{0,1\}^\ell$, it holds that $\{\mathbf{G}(\lambda, c, 1)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{H}_1(\lambda, c)\}_{\lambda\in\mathbb{N}}$.*

Proof. The proof is down to the (computational) SHVZK property of Σ. By contradiction, assume that there exists $c \in \{0,1\}^\ell$, a PPT attacker $\mathcal{A}''$, and a polynomial $p(\lambda)$ such that

$$|\mathbb{P}[\mathbf{G}(\lambda, c, 1) = 1] - \mathbb{P}[\mathbf{H}_1(\lambda, c) = 1]| \geq 1/p(\lambda).$$

We are going to construct a PPT attacker $\mathcal{A}$ breaking the SHVZK property of Σ w.r.t. challenge c. A description of $\mathcal{A}$ follows:

- At the outset pick $(T, \alpha) \leftarrow\!\!\$\ R_{DH}$, run $(\mathtt{com}, \mathtt{rand}) \leftarrow\!\!\$\ \mathsf{Fake}_1(T, \alpha)$ and $(a', z') \leftarrow\!\!\$\ \mathcal{S}'(T, c)$, and forward $a'' = (a', \mathtt{com}, T)$ and c to $\mathcal{A}''$.
- Upon receiving (x, w) from $\mathcal{A}$:
  - In case $(x, w) \in R$, forward x to the challenger obtaining (a, z). Hence, let $\mathtt{dec} \leftarrow\!\!\$\ \mathsf{Fake}_2((T, \alpha), a, (\mathtt{com}, \mathtt{rand}))$ and forward $z'' = (z, z', (\mathtt{dec}, a))$ to $\mathcal{A}''$.
  - In case $(x, w) \notin R$, forward $z'' = \perp$ to $\mathcal{A}''$.
- Output whatever $\mathcal{A}''$ outputs.

[10] All games are further parameterized by an attacker and the protocol Σ'' of Fig. 1, but we avoid to write this explicitly to simplify notation.

For the analysis, note that the distribution of a'' as sampled by the reduction is identical to that of those values in both experiments $\mathbf{G}(\lambda, c, 1)$ and $\mathbf{H}_1(\lambda, c)$. The same holds for the distribution of z'' whenever $(x, w) \notin R$. As for the value z'', in case (a, z) are sampled using the prover in Σ (*i.e.*, $(a, st) \leftarrow\!\!\$\ \mathcal{P}(1^\lambda)$ and $z \leftarrow\!\!\$\ \mathcal{P}(x, w, a, c)$) the distribution of z'' is identical to that in experiment $\mathbf{H}_1(\lambda, c)$, whereas in case (a, z) are sampled using the SHVZK simulator in Σ (*i.e.*, $(a, z) \leftarrow\!\!\$\ \mathcal{S}(x, c)$) the distribution of $z''$is identical to that in experiment $\mathbf{G}(\lambda, c, 1)$. Hence, the distinguishing advantage of $\mathcal{A}$ is identical to that of $\mathcal{A}''$. The statement follows. □

Claim 2. *For all $c \in \{0,1\}^\ell$, it holds that $\{\mathbf{H}_1(\lambda, c)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{H}_2(\lambda, c)\}_{\lambda\in\mathbb{N}}$.*

Proof. The proof is down to the DDH assumption. By contradiction, assume that there exists $c \in \{0,1\}^\ell$, a PPT attacker $\mathcal{A}''$, and a polynomial $p(\lambda)$ such that

$$|\mathbb{P}[\mathbf{H}_1(\lambda, c) = 1] - \mathbb{P}[\mathbf{H}_2(\lambda, c) = 1]| \geq 1/p(\lambda).$$

We are going to construct a PPT attacker $\mathcal{A}$ breaking the DDH assumption. A description of $\mathcal{A}$ follows:

- At the outset, receive a tuple $T = (g, A, B, C)$ from the challenger.
- Run $(a, st) \leftarrow\!\!\$\ \mathcal{P}(1^\lambda)$, $(\mathtt{com}, \mathtt{dec}) \leftarrow\!\!\$\ \mathsf{Com}(T, a)$, and $(a', z') \leftarrow\!\!\$\ \mathcal{S}'(T, c)$, and forward $a'' = (\mathtt{com}, a', T)$ and c to $\mathcal{A}''$.
- Upon receiving (x, w) from $\mathcal{A}$:
  - In case $(x, w) \in R$, let $z \leftarrow\!\!\$\ \mathcal{P}(x, w, c, st)$ and forward $z'' = (z, z', (\mathtt{dec}, a))$ to $\mathcal{A}''$.
  - In case $(x, w) \notin R$, forward $z'' = \perp$ to $\mathcal{A}''$.
- Output whatever $\mathcal{A}''$ outputs.

For the analysis, note that the reduction runs exactly the same code as in $\mathbf{H}_2(\lambda, c)$ after embedding the tuple T from the challenger. In particular, assuming that T is a 1-non-DH tuple, the reduction perfectly emulates the view of $\mathcal{A}''$ in $\mathbf{H}_2(\lambda, c)$.

On the other hand, assuming that T is a DH tuple, we claim that the reduction perfectly emulates the view of $\mathcal{A}''$ in $\mathbf{H}_1(\lambda, c)$. The latter follows by the perfect trapdoorness property of Π, as for all $(T, \alpha) \in R_{DH}$, and for every message a, the distribution of $(\mathtt{com}, \mathtt{dec})$ in $(\mathtt{com}, \mathtt{dec}) \leftarrow\!\!\$\ \mathsf{Com}(T, a)$ is identical to the distribution of $(\mathtt{com}, \mathtt{dec})$ in $(\mathtt{com}, \mathtt{rand}) \leftarrow\!\!\$\ \mathsf{Fake}_1(T, \alpha)$ and $\mathtt{dec} \leftarrow\!\!\$\ \mathsf{Fake}_2((T, \alpha), a, (\mathtt{com}, \mathtt{rand}))$. Hence, the distinguishing advantage of $\mathcal{A}$ is identical to that of $\mathcal{A}''$. The statement follows. □

Claim 3. *For all $c \in \{0,1\}^\ell$, it holds that $\{\mathbf{H}_2(\lambda, c)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{G}(\lambda, c, 0)\}_{\lambda\in\mathbb{N}}$.*

Proof. The proof is down to the (computational) SHVZK property of Σ'. By contradiction, assume that there exists $c \in \{0,1\}^\ell$, a PPT attacker $\mathcal{A}''$, and a polynomial $p(\lambda)$ such that

$$|\mathbb{P}[\mathbf{H}_2(\lambda, c) = 1] - \mathbb{P}[\mathbf{G}(\lambda, c, 0) = 1]| \geq 1/p(\lambda).$$

We are going to construct a PPT attacker $\mathcal{A}'$ breaking the SHVZK property of Σ' w.r.t. challenge c. A description of $\mathcal{A}'$ follows:

- At the outset pick $(T, \alpha) \leftarrow\!\!\$\ R_{1nDH}$, and forward (T, α) to the challenger obtaining a transcript (a', c, z').
- Compute $(a, st) \leftarrow\!\!\$\ \mathcal{P}(1^\lambda)$, let $(\mathtt{com}, \mathtt{dec}) \leftarrow\!\!\$\ \mathsf{Com}(T, a)$ and forward $a'' = (a', \mathtt{com}, T)$ and c to $\mathcal{A}''$.
- Upon receiving (x, w) from $\mathcal{A}$:
  - In case $(x, w) \in R$, let $z \leftarrow\!\!\$\ \mathcal{P}(x, w, a, c)$ and forward $z'' = (z, z', (\mathtt{dec}, a))$ to $\mathcal{A}''$.
  - In case $(x, w) \notin R$, forward $z'' = \perp$ to $\mathcal{A}''$.
- Output whatever $\mathcal{A}''$ outputs.

For the analysis, note that in case (a', z') are sampled using the SHVZK simulator in Σ' (*i.e.*, $(a', z') \leftarrow_{\$} \mathcal{S}'(T, c)$) the reduction perfectly emulates the view of $\mathcal{A}''$ in $\mathbf{H}_2(\lambda, c)$. On the other hand, in case (a', z') are sampled using the prover in Σ' (*i.e.*, $(a', st') \leftarrow_{\$} \mathcal{P}'(T, \alpha)$ and $z' \leftarrow_{\$} \mathcal{P}'(st', c)$) the reduction perfectly emulates the view of $\mathcal{A}''$ in $\mathbf{G}(\lambda, c, 0)$. Hence, the distinguishing advantage of $\mathcal{A}'$ is identical to that of $\mathcal{A}''$. The statement follows. □

Lemma 3 follows by combining the above claims. □

The theorem follows by combining Lemma 2 and Lemma 3. □

4 Adaptive Security of the Fiat-Shamir Transform

4.1 Trapdoor Sigma Protocols and the Fiat-Shamir Transform

Informally, a trapdoor Sigma protocol is a special Sigma protocol in the CRS model satisfying the following two properties: (i) If the statement x is false, then for every first message a, there is a unique challenge c for which there is an accepting third message z that results in an accepting transcript (a, c, z); (ii) There is a trapdoor associated with the CRS that allows us to efficiently compute this "bad challenge" c from the first message a and the statement x being proven. We now slightly revisit the definition of trapdoor Sigma protocols from [CCH+19], and show that the Fiat-Shamir transform applied to a trapdoor Sigma protocol (where the hash function takes as input both the statement and the first round of the prover) yields a NIZK with *adaptive soundness*. The only difference between the definition in [CCH+19] and ours is that we require the honestly generated CRS to be identically distributed to the CRS generated together with the trapdoor.[11] We also show that, assuming the trapdoor Sigma protocol admits an adaptive-input SHVZK simulator, then the NIZK resulting from the FS transform (where the hash function now takes as input only the first round of the prover) satisfies *adaptive zero knowledge*.

Definition 14 (CRSigma protocols)**.** We say that a three-round public-coin SHVZK proof system $\Sigma = (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$[12] in the CRS model is a CRSigma protocol if for every valid CRS ω, every instance $x \notin L$, and every first round a, there is at most one challenge $c := f(\omega, x, a)$ such that (ω, x, a, c, z) is an accepting transcript for some z. We informally call f the "bad-challenge function" associated to Σ, and note that f may not be efficiently computable.[13]

Following [CCH+19], we now define trapdoor Sigma protocol to be a CRSigma protocol that has a trapdoor making the bad-challenge function f efficiently computable.

Definition 15 (Trapdoor Sigma protocol)**.** We say that a CRSigma protocol $\Sigma = (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ with bad-challenge function f is a *trapdoor Sigma protocol* if there are PPT algorithms $\mathsf{TrapGen}$, $\mathsf{BadChallenge}$ with the following syntax:

- $\mathsf{TrapGen}$ takes as input the unary representation of the security parameter and outputs a common reference string ω with a trapdoor τ.
- $\mathsf{BadChallenge}$ takes as input a trapdoor τ, common reference string ω, an instance x and the first message a and outputs a challenge c.

We additionally require the following properties.

- **CRS indistinguishability.** An honestly generated common reference string ω is identically distributed to a common reference string output by $\mathsf{TrapGen}(1^\lambda)$.
- **Correctness.** For every instance $x \notin L$ and for all $(\omega, \tau) \leftarrow_{\$} \mathsf{TrapGen}(1^\lambda)$ we have that $\mathsf{BadChallenge}(\tau, \omega, x, a) = f(\omega, x, a)$.

[11] This modification is related to the fact that we want to prove adaptive soundness (more on this later).
[12] In this case the SHVZK simulator computes also the CRS.
[13] We observe that this notion implies that a trapdoor Sigma protocol is sound with soundness error $2^{-|c|}$.

The Fiat-Shamir transform. Let $\mathcal{H}$ be a hash family and $\Sigma = (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ be a (delayed-input) CRSigma protocol for some relation R. Consider the following non-interactive argument systems $\Pi' = (\mathsf{Gen}', \mathcal{P}', \mathcal{V}')$ and $\Pi'' = (\mathsf{Gen}', \mathcal{P}'', \mathcal{V}')$ for R:

- The common reference string $\omega' := (\omega, k)$ consists of the common reference string of Σ (*i.e.*, $\omega \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$) along with a hash key $k \leftarrow\!\!\$\ \mathcal{H}.\mathsf{Gen}(1^\lambda)$.
- Upon input $(x, w) \in R$, the prover $\mathcal{P}'$ (resp. $\mathcal{P}''$) computes $(a, st) \leftarrow\!\!\$\ \mathcal{P}(1^\lambda, \omega, x, w)$ (resp. $(a, st) \leftarrow\!\!\$\ \mathcal{P}(1^\lambda, \omega)$), $c := h_k(a||x)$ (resp. $c := h_k(a)$) and $z \leftarrow\!\!\$\ \mathcal{P}(st, c)$ (resp. $z \leftarrow\!\!\$\ \mathcal{P}(st, x, w, c)$), and outputs[14] (a, c, z).
- The verifier $\mathcal{V}'$ (resp. $\mathcal{V}''$) accepts the transcript (a, c, z) w.r.t. CRS $\omega' = (\omega, k)$ and statement x if $\mathcal{V}(\omega, x, a, c, z) = 1$ and $h_k(a||x) = c$ (resp. $h_k(a) = c$).

Theorem 4. *Suppose that $\mathcal{H}$ is a hash family that is correlation intractable for all sub-exponentially sparse relations that are searchable in time t, and that $\mathcal{H}$ enjoys programmability. Moreover, assume that $\Sigma = (\mathsf{Gen}, \mathcal{P}, \mathcal{V}, \mathsf{TrapGen}, \mathsf{BadChallenge})$ is a trapdoor Sigma protocol with SHVZK and challenge (second message) space $\{0,1\}^{\lambda^\epsilon}$ for some $\epsilon > 0$, such that $\mathsf{BadChallenge}(\tau, \omega, x, a)$ is computable in time t. Then, the non-interactive argument system Π' described above satisfies zero-knowledge and* adaptive soundness *in the CRS model.*

Proof. We prove adaptive soundness and (non-adaptive) zero knowledge of Π' separately.

Adaptive soundness. By construction, and by the definition of trapdoor Sigma protocol, we know that for every $x \notin L$ and every ω, an accepting transcript for Π' must satisfy the condition that $h_k(a||x) = c = f(\omega, x, a)$. Suppose now that some PPT prover $P^*(\omega, k)$ outputs, with non-negligible probability, a false statement $x \notin L$ and an accepting proof (a, c, z). Then, by (perfect) CRS indistinguishability, the same would be true for ω sampled by the algorithm $\mathsf{TrapGen}(1^\lambda)$ which also outputs the trapdoor τ. In this case, P^* would still output (with non-negligible probability) some x, a such that $h_k(a||x) = f(\omega, x, a) = \mathsf{BadChallenge}(\tau, \omega, x, a)$. This contradicts the correlation intractability of $\mathcal{H}$ for the relation $R_{\omega,\tau} := \{((a||x), c) : c = \mathsf{BadChallenge}(\tau, \omega, x, a) \text{ and } x \notin L\}$.

In more details, an adversary $\mathcal{A}$ could break correlation intractability of $\mathcal{H}$ by sampling (ω, τ) itself, declaring the relation $R_{\omega,\tau}$ to be broken, and then running $\mathcal{P}^*(\omega, k)$ after being given $k \leftarrow\!\!\$\ \mathcal{H}.\mathsf{Gen}(1^\lambda)$. Since Σ has challenge space $\{0,1\}^{\lambda^\epsilon}$, the relation $R_{\omega,\tau}$ indeed has sub-exponential sparsity. Moreover, if $((a||x), c)$ is in $R_{\omega,\tau}$ then c is efficiently computable given τ, and thus $R_{\omega,\tau}$ is efficiently searchable, which contradicts our assumption on $\mathcal{H}$. This concludes the proof of adaptive soundness.

Zero knowledge. The simulator $\mathcal{S}$, upon input the instance $x \in L$, works as follows.

- Pick a random challenge c for Σ.
- Run the SHVZK simulator of Σ upon input (x, c), thus obtaining ω, a, z.
- Sample a hash key k from the conditional distribution $k \leftarrow\!\!\$\ \mathcal{H}.\mathsf{Gen}(1^\lambda)$ s.t. $h(k, a||x) = c$.
- Output (ω, k, a, c, z)

Indistinguishability of the simulation follows from the programmability of the hash function, and by the SHVZK property of Σ. This concludes the proof of (non-adaptive) zero knowledge. □

Theorem 5. *Suppose that $\mathcal{H}$ is a hash family that is correlation intractable for all sub-exponentially sparse relations that are searchable in time t, and that $\mathcal{H}$ enjoys programmability. Moreover,*

[14] Equivalently, the prover can just output (a, z) as c can be re-computed by the verifier.

assume that $\Sigma = (\mathsf{Gen}, \mathcal{P}, \mathcal{V}, \mathsf{TrapGen}, \mathsf{BadChallenge})$ *is a trapdoor Sigma protocol with* adaptive-input *SHVZK*[15] *and challenge space* $\{0,1\}^{\lambda^\epsilon}$ *for some* $\epsilon > 0$*, such that* $\mathsf{BadChallenge}(\tau, \omega, x, a)$ *is computable in time* t*. Then, the non-interactive argument system* Π'' *described above satisfies soundness and* adaptive zero knowledge *in the CRS model.*

Proof. We show (non-adaptive) soundness[16] and adaptive zero knowledge of Π'' separately.

Soundness. By construction, and by the definition of trapdoor Sigma protocols, we know that for every $x \notin L$ and every ω, an accepting transcript for Π'' must satisfy the condition that $h_k(a) = c = f(\omega, x, a)$. Suppose that there exists a theorem $x \notin L$ and some PPT prover $\mathcal{P}^*$ such that $P^*(\omega, k)$ outputs, with non-negligible probability, an accepting proof (a, c, z). Then, by (perfect) CRS indistinguishability, the same would be true for ω sampled by the algorithm $\mathsf{TrapGen}(1^\lambda)$ which also outputs the trapdoor τ. In this case, P^* would still output (with non-negligible probability) an a such that $h_k(a) = f(\omega, x, a) = \mathsf{BadChallenge}(\tau, \omega, x, a)$. This contradicts the correlation intractability of $\mathcal{H}$ for the relation $R_{\omega,\tau,x} := \{(a, c) : c = \mathsf{BadChallenge}(\tau, \omega, x, a) \text{ and } x \notin L\}$.

In more details, an adversary $\mathcal{A}$ could break correlation intractability of $\mathcal{H}$ by sampling (ω, τ) itself, declaring the relation $R_{\omega,\tau,x}$ to be broken, and then running $\mathcal{P}^*((\omega, k), x)$ after being given $k \leftarrow\!\!\$\ \mathcal{H}.\mathsf{Gen}(1^\lambda)$. Since Σ has challenge space $\{0,1\}^{\lambda^\epsilon}$, the relation $R_{\omega,\tau,x}$ indeed has sub-exponential sparsity, which contradicts our assumption on $\mathcal{H}$. This concludes the proof of (non-adaptive) soundness.

Adaptive zero knowledge. Recall that we assume that Σ has an *adaptive-input* SHVZK simulator $(\mathcal{S}_0, \mathcal{S}_1)$. The simulator $\mathcal{S} := (\mathcal{S}''_0, \mathcal{S}''_1)$ for Π'' works as follows.

- $\mathcal{S}''_0(1^\lambda, 1^{|x|})$:
  - Pick a random challenge c for Σ.
  - Run $\mathcal{S}'_1$ on input 1^λ,$1^{|x|}$ and c, thus obtaining st, ω, a.
  - Sample a hash key k from the conditional distribution $k \leftarrow\!\!\$\ \mathcal{H}.\mathsf{Gen}(1^\lambda)$ s.t. $h_k(a) = c$.
  - Output $\omega'' := (\omega, k)$ and $st'' := (st, a)$.
- $\mathcal{S}''_1(st'', x)$:
  - Run $\mathcal{S}_1$ upon input st, x, thus obtaining z.
  - Output (a, c, z).

Indistinguishability of the simulation follows from the programmability of the hash function and the adaptive-input SHVZK property of Σ. This concludes the proof of adaptive zero knowledge.

□

4.2 From CRSigma Protocols to Trapdoor Sigma Protocols

In [CCH+19], the authors show that a modified version of the protocol for Hamiltonian graphs [FLS90, LS91] is a trapdoor Sigma protocol. This allows to obtain a trapdoor Sigma protocol for any NP relation R by just making an NP reduction. In this section we show that *any* CRSigma protocol can be turned into a trapdoor Sigma protocol without making use of expensive NP reductions. Let $\Sigma = (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ be a CRSigma protocol for a polynomial-time relation R. Without loss of generality, we assume that the challenge space of Σ is $\{0,1\}$. We construct a trapdoor Sigma protocol $\Sigma' := (\mathsf{Gen}', \mathcal{P}', \mathcal{V}')$ for R based on Σ and on a public-key encryption

[15] As in the definition of adaptive-input SHVZK, the simulator is defined by two algorithms $(\mathcal{S}_0, \mathcal{S}_1)$. The difference is that $\mathcal{S}_0$ outputs the CRS in addition.

[16] The proof of soundness is almost identical to that of [CCH+19, Theorem 6.4]. For the sake of completeness, we repeat it here.

Common input: Security parameter λ, instance $x \in L$.
Prover's private input: Witness w s.t. $(x, w) \in R$.
CRS generation: Run $(pk, sk) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$ and $\omega \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$, and set $\omega' := (pk, \omega)$.
$\mathcal{P}'(x, w)$**:** The first prover's message is generated as follows.

- Compute $(a, st) \leftarrow\!\!\$\ \mathcal{P}(x, w)$, $z_0 \leftarrow\!\!\$\ \mathcal{P}(st, 0)$, $z_1 \leftarrow\!\!\$\ \mathcal{P}(st, 1)$.
- Pick $r_0 \leftarrow\!\!\$\ \{0,1\}^\lambda$ and run Enc with randomness r_0 on input pk and z_0, thus obtaining e_0.
- Pick $r_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$ and run Enc with randomness r_1 on input pk and z_1, thus obtaining e_1.
- Send $a' := (e_0, e_1, a)$ to $\mathcal{V}'$ and let $st' = (r_0, r_1, z_0, z_1)$.

$\mathcal{V}'(1^\lambda)$**:** Pick $c \leftarrow\!\!\$\ \{0,1\}$ and send it to $\mathcal{P}'$.
$\mathcal{P}'(st', c)$**:** Send $z := (z_c, r_c)$ to $\mathcal{V}'$.
Verifier's verdict: Accept if and only if (i) $\mathcal{V}(\omega, x, a, c, z_c) = 1$; and (ii) $\mathsf{Enc}(pk, z_c; r_c) = e_c$.

Figure 3: Our compiler: from the CRSigma protocol Σ to the trapdoor Sigma protocol Σ'.

(PKE) scheme $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ with perfect correctness. The PKE scheme is essentially used as a commitment, similarly to what is done in [CCH+19]. At a high level our transform works as follows. The CRS consists of a public key for the PKE scheme and of a CRS for Σ. To compute a proof, the prover generates the first message of Σ and the replies to the challenge 0 and 1 that we denote respectively with z_0 and z_1. Then, the prover encrypts z_0 and z_1 and sends these encrypted values together with the first round of Σ to the verifier. The verifier sends a random bit c, and the prover replies with z_c and the randomness used to compute the encryption of z_c. Finally, the verifier accepts if the randomness and the value z_c are consistent with the commitment received in the first round and if the transcript for Σ is accepting. The formal description appears in Fig. 3. We note that given the secret key of the encryption scheme it is possible to extract the bad challenge (if any). And this is the intuitive reason why the protocol of Fig. 3 is a trapdoor Sigma protocol. More formally, we can prove the following theorem.

Theorem 6. *If $\Sigma := (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ is a CRSigma protocol for the NP relation R and $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is a secure PKE scheme with perfect correctness, then the protocol $\Sigma' := (\mathsf{Gen}', \mathcal{P}', \mathcal{V}')$ of Fig. 3 is a trapdoor Sigma protocol.*

Proof. We start by describing the algorithms $\mathsf{TrapGen}$ and $\mathsf{BadChallenge}$, and then we show that Σ enjoys the properties of CRS indistinguishability and correctness.

- Algorithm $\mathsf{TrapGen}$ acts exactly as the CRS generation procedure of Σ. That is, $\mathsf{TrapGen}(1^\lambda)$ runs $\omega \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$ and $(pk, sk) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$, and sets $\omega' := (pk, \omega)$ and $\tau := sk$. This immediately gives perfect CRS indistinguishability.
- Algorithm $\mathsf{BadChallenge}$ works as follows. Upon input $(\tau := sk, \omega' := (pk, \omega), x, a' := (e_0, e_1, a))$, it computes $z_0 = \mathsf{Dec}(sk, e_0)$ and $z_1 = \mathsf{Dec}(sk, e_1)$. Finally, if there exists c such that $\mathcal{V}(\omega, x, a, c, z_c) = 1$ it outputs c, and otherwise it outputs $\perp$.

We now argue that, if $x \notin L$, then the output of $\mathsf{BadChallenge}$ is identical to that of the bad-challenge function $f(\omega', x, a')$. We can consider the following cases:

(i) $\mathcal{V}(\omega, x, a, 0, z_0) = 1$ and $\mathcal{V}(\omega, x, a, 1, z_1) = 1$; or
(ii) $\exists! c \in \{0,1\}$ such that $\mathcal{V}(\omega, x, a, c, z) = 1$; or
(iii) none of the above occurs.

Case (i) cannot happen, as this would contradict the fact that Σ is a CRSigma protocol. Indeed, for any $x \notin L$ and first round a of a CRSigma protocol there exists only one challenge that would make the verifier to accept. Case (iii) is not interesting, as in this case there is no bad challenge to extract (this is guaranteed by perfect correctness of the PKE scheme). Case (ii) implies the correctness property.

Finally, it remains[17] to show that Σ' satisfies the SHVZK property. Let $\mathcal{S}$ be the SHVZK simulator of Σ. The simulator $\mathcal{S}'$ proceeds as follows. Upon input (x, c), simulator $\mathcal{S}'$ runs $\mathcal{S}(x, c)$

[17] Recall that the definition of trapdoor Sigma protocols automatically implies that Σ' has soundness error $1/2$.

thus obtaining (ω, a, z); hence, it sets $\omega' := (pk, \omega)$ and $a' := (e_0, e_1, a)$ for a freshly generated pair of keys $(pk, sk) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$, and where the ciphertext e_c is obtained as an encryption of z (*i.e.*, $e_c := \mathsf{Enc}(pk, z; r_c)$ for $r_c \leftarrow\!\!\$\ \{0,1\}^\lambda$), whereas the ciphertext e_{1-c} is obtained as an encryption of the all-zero string (*i.e.*, $e_{1-c} := \mathsf{Enc}(pk, 0^{|z|}; r_{1-c})$ for $r_{1-c} \leftarrow\!\!\$\ \{0,1\}^\lambda$). Finally, $\mathcal{S}'$ outputs (ω', a', z'), where $z' := (z_c, r_c)$. Indistinguishability of the simulation follows immediately by semantic security of the PKE scheme. This finishes the proof. □

We remark that it is always possible to extend the challenge space of the above protocol to $\{0,1\}^\kappa$ for any $\kappa \in \mathbb{N}$ without compromising its completeness, by just repeating it in parallel κ times. Then, using Theorem 4, we obtain an *adaptively-sound* NIZK.

5 Adding Adaptive-Input SHVZK

To transform a delayed-input trapdoor Sigma protocol $\Sigma = (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ into one with adaptive-input SHVZK we follow the same approach proposed in §3. The prover computes the first round of Σ and commits to it using an IDTC. Upon receiving a random challenge from the verifier, the prover opens the commitment and sends the third round of Σ. In addition, a Sigma protocol to prove that the IDTC is computed using a false instance (*i.e.*, a non-DH tuple) is run in parallel.

Unfortunately, it is not clear whether the output Σ'' of such a compiler would still be a trapdoor Sigma protocol. Indeed, an IDTC does not provide any extractability property. Hence, given the first round of the above protocol (which consists of a commitment to the first round of Σ), it is not clear how to extract the committed value. We note that if it was possible to extract the committed value then we would be almost done, as then we can rely on algorithm $\mathsf{BadChallenge}$ of Σ to obtain the bad challenge. To tackle this problem, we introduce a new primitive which we dub *extractable IDTC*. An extractable IDTC is similar to an IDTC but it is equipped with a setup algorithm that generates a CRS and a trapdoor. Given the trapdoor it is possible to extract the message committed via the extractable IDTC w.r.t. a false instance (*i.e.*, in case the commitment is binding). In this section we provide a generic compiler that turns any delayed-input trapdoor Sigma protocol into one with *adaptive-input* SHVZK. We first start by describing formally the tools required by our compiler.

5.1 The Sigma Protocol for DH Tuples

Let us recall the following well-known Sigma protocol $\Sigma_{DH} = (\mathcal{P}, \mathcal{V})$ for relation R_{DH}. On common input $T = (g, h, U, V)$, and honest prover's private input α such that $U = g^\alpha$ and $V = h^\alpha$, the following steps are executed. We denote the size of the group $\mathcal{G}$ by q.

- $\mathcal{P}$ picks $r \in \mathbb{Z}_q$ at random and computes and sends $A := g^r$, $B := h^r$ to $\mathcal{V}$;
- $\mathcal{V}$ chooses a random challenge $c \in \{0,1\}$ and sends it to $\mathcal{P}$;
- $\mathcal{P}$ computes and sends $z = r + \alpha \cdot c$ to $\mathcal{V}$;
- $\mathcal{V}$ accepts if and only if $g^z = A \cdot U^c$ and $h^z = B \cdot V^c$.

The above protocol has the following interesting property. There exists a PPT algorithm $\mathsf{ChallExt}$ that on input a (potentially maliciously generated) first round $a = (A, B)$ of Σ_{DH}, a non-DH tuple T and γ such that $h = g^\gamma$, outputs the only valid second round $c \in \{0,1\}$ (if any exists) such that there is some z that would make the verifier to (mistakenly) accept the transcript (a, c, z) with respect to the instance T.

The algorithm $\mathsf{ChallExt}$ works as follows. Let $T = (g, h, X, W)$ be a non-DH tuple such that $X = g^\alpha$, $W = h^\beta$, $\alpha \neq \beta$ and $h = g^\gamma$. Upon input $(T = (g, h, X, W), a, \gamma)$, algorithm $\mathsf{ChallExt}$ parses a as (A, B), and if $A^\gamma = B$ then it outputs 0, else it outputs 1.

Note that when the first round of Σ_{DH} corresponds to a DH tuple, (*i.e.*, $A^\gamma = B$) and T is not a DH tuple, then the only c that would make true the conditions $g^z = A \cdot U^c$ and $h^z = B \cdot V^c$

is $c = 0$. Instead, if (g, h, A, B) does not represent a DH tuple (*i.e.*, $A^\gamma \neq B$) then there exists z such that $g^z = A \cdot U^c$ and $h^z = B \cdot V^c$ if and only if $c = 1$.

In what follows, we make use of this special property of Σ_{DH}, and we refer to $\mathsf{ChallExt}$ as the *bad-challenge extractor*. It is easy to see that also the Sigma protocol for the NP relation R_{1nDH} (see §2.4) admits such an extractor $\mathsf{ChallExt}$. The same holds true for the classical Sigma protocol for QR [GMR89] (along the lines of the full version of [CCH+19, §6.2]).

5.2 Extractable IDTCs

In this section, we consider a slight generalization of IDTCs (see § 2.5) that we call *extractable IDTCs*. An extractable IDTC is still defined with respect to an NP language L, but has the following differences with respect to an IDTC.

1. It is defined in the CRS model. That is, there is a generation algorithm that outputs a CRS ω and a trapdoor τ (possibly depending on L).
2. The algorithms used to compute a commitment (either in binding or in trapdoor mode), do not take the instance as input. Instead, the instance is part of the output of the commitment procedure.
3. It has an extractor Ext that on input a commitment computed w.r.t. a false instance (*i.e.*, using the binding mode), and the trapdoor τ, outputs the committed message.

Formally an extractable IDTC scheme for polynomial-time relation R (and corresponding NP language L) with message space M is a tuple of PPT algorithms $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Dec}, \mathsf{Fake}_1, \mathsf{Fake}_2, \mathsf{Ext})$ specified as follows:

- Gen is a randomized algorithm that takes as input the security parameter and outputs a CRS ω and a trapdoor τ.
- Com is the randomized *commitment* algorithm that takes as input the CRS ω and a message $m \in M$ and outputs $(\mathtt{com}, T, \alpha, \mathtt{dec})$ such that $T \in \overline{L}$ and $(T, \alpha) \in R_{\overline{L}}$. We refer to $\mathtt{com}$ as the commitment and to $\mathtt{dec}$ as the decommitment information.
- Dec is the *verification* algorithm that takes as input $\omega, \mathtt{com}, T, \mathtt{dec}$ and $m \in M$ and decides whether m is the decommitment of $\mathtt{com}$ with respect to the instance T and the CRS ω;
- Fake_1 takes as input ω and outputs the *commitment* $\mathtt{com}$, an instance T such that $T \in L$ and the *equivocation information* $\mathtt{rand}$;
- Fake_2 takes as input a message $m \in M$, the tuple $(\mathtt{com}, T, \mathtt{rand})$ and outputs $\mathtt{dec}$;
- Ext takes as input ω, the commitment $\mathtt{com}$ with respect to the instance T, and the trapdoor τ and outputs m.

Definition 16 (Extractable IDTC scheme)**.** We call $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Dec}, \mathsf{Fake}_1, \mathsf{Fake}_2, \mathsf{Ext})$ an *extractable IDTC* scheme for the NP relation R (and corresponding NP language L) if it enjoys the following properties:

- **Correctness.** For all $m \in M$, it holds that

$$\mathbb{P}\left[\mathsf{Dec}(\omega, \mathtt{com}, T, \mathtt{dec}, m) = 1 : \ (\mathtt{com}, T, \mathtt{dec}) \leftarrow\!\!\$\ \mathsf{Com}(\omega, m)\right] = 1.$$

- **Binding.** For all $T \notin L$, and for every commitment $\mathtt{com}$, there exists at most one message $m \in M$ for which there is a valid decommitment $\mathtt{dec}$ (*i.e.*, $\mathsf{Dec}(\omega, \mathtt{com}, T, \mathtt{dec}, m) = 1$).
- **Trapdoorness.** For all $m \in M$ and honestly generated ω (*i.e.*, computed running Gen) the following two distributions coincide:

$$\{(\omega, \mathtt{com}, T, \mathtt{dec}, m) : \ (\mathtt{com}, T, \mathtt{rand}) \leftarrow\!\!\$\ \mathsf{Fake}_1(\omega); \mathtt{dec} \leftarrow\!\!\$\ \mathsf{Fake}_2(\omega, \mathtt{com}, m, \mathtt{rand})\}$$
$$\{(\omega, \mathtt{com}, T, \mathtt{dec}, m) : \ (\mathtt{com}, T, \mathtt{dec}) \leftarrow\!\!\$\ \mathsf{Com}(\omega, m)\}\,.$$

- **Extractability.** For all $\mathtt{com} \in \{0,1\}^{poly(\lambda)}$ and T with $T \notin L$, if there exists $(m, \mathtt{dec})$ such that $\mathsf{Dec}(\omega, \mathtt{com}, T, \mathtt{dec}, m) = 1$ then it holds that

$$\mathbb{P}\left[\mathsf{Ext}(\omega, \mathtt{com}, T, \tau) = m\right] = 1.$$

5.2.1 The Basic Construction

We now show how to obtain an extractable IDTC $\Pi^{\text{1-bit}} = (\mathsf{Gen}^{\text{1-bit}}, \mathsf{Com}^{\text{1-bit}}, \mathsf{Dec}^{\text{1-bit}}, \mathsf{Fake}_1^{\text{1-bit}}, \mathsf{Fake}_2^{\text{1-bit}}, \mathsf{Ext}^{\text{1-bit}})$ for the NP-relation R_{DH} and with messages space $\{0,1\}$ assuming the hardness of the DDH assumption and using as the main tool the Sigma protocol $\Sigma_{DH} = (\mathcal{P}, \mathcal{V})$ (described in §5.1) equipped with its SHVZK simulator $\mathcal{S}$.

- $\mathsf{Gen}^{\text{1-bit}}$ runs the group generator GG (see §2.4), thus obtaining g, q, and outputs (ω, τ) with $\omega := (g, h)$ and $h = g^\tau$ for a random $\tau \in \mathbb{Z}_q$.
- $\mathsf{Com}^{\text{1-bit}}$ takes as input $\omega = (g, h)$ and the message $m \in \{0,1\}$, picks $\alpha \leftarrow\!\!\$\ \mathbb{Z}_q$, computes a 1-non-DH tuple $T = (g, h, g^\alpha, gh^\alpha)$ and $(a, z) \leftarrow\!\!\$\ \mathcal{S}(T, m)$. The output of $\mathsf{Com}^{\text{1-bit}}$ corresponds to $\mathtt{com} := a$, T, α and $\mathtt{dec} := z$.
- $\mathsf{Dec}^{\text{1-bit}}$ takes as input $\omega := (g, h)$, $\mathtt{com} := a$, T, $\mathtt{dec} := z$ and m, parses T as (g', h', X, W) and accepts if and only if $\mathcal{V}(T, a, m, z) = 1$ and $\omega = (g', h')$.
- $\mathsf{Fake}_1^{\text{1-bit}}$ takes as input $\omega = (g, h)$, picks $\alpha \leftarrow\!\!\$\ \mathbb{Z}_q$ and computes a DH tuple $T = (g, h, g^\alpha, h^\alpha)$ and $(a, st) \leftarrow\!\!\$\ \mathcal{P}(T, \alpha)$. The output of $\mathsf{Com}^{\text{1-bit}}$ corresponds to $\mathtt{com} := a$, T, $\mathtt{rand} := st$.
- $\mathsf{Fake}_2^{\text{1-bit}}$ takes as input ω, the message $m \in \{0,1\}$, $\mathtt{com} := a, \mathtt{rand} := st$ and computes $z \leftarrow\!\!\$\ \mathcal{P}(st, m)$. The output of $\mathsf{Fake}_2^{\text{1-bit}}$ corresponds to z, m.
- $\mathsf{Ext}^{\text{1-bit}}$ takes as input $(\omega, \mathtt{com} := a, T, \tau)$ and runs $\mathsf{ChallExt}(T, \mathtt{com}, \tau)$ thus obtaining m. The output of $\mathsf{Ext}^{\text{1-bit}}$ corresponds to m.

Theorem 7. *Assuming that the DDH assumption holds, then* $\Pi^{\text{1-bit}}$ *is an extractable IDTC with message space* $\{0,1\}$.

Proof. We proceed to show each of the required properties in turn.

- **Correctness.** Correctness follows by inspection.
- **Binding.** This property follows immediately from the special soundness of $\Pi^{\text{1-bit}}$. Indeed, the output of Com corresponds always to the first round of $\Pi^{\text{1-bit}}$ computed with respect to a false statement T (where T is a 1-non-DH tuple). Hence, there exists only one valid second round m such that $\mathcal{V}(T, a, m, z) = 1$ for some z.
- **Trapdoorness.** The proof follows via a standard hybrid argument. In the first hybrid experiment $\mathbf{H}_1$ the commitment and the decommitment is computed using the algorithm Com. The second hybrid experiment $\mathbf{H}_2$ differs from $\mathbf{H}_1$ in the way the tuple T is computed. Indeed, in this case T is set to be a DH tuple. The indistinguishability between $\mathbf{H}_1$ and $\mathbf{H}_2$ comes immediately from the hardness of the DDH assumption. In the hybrid experiment $\mathbf{H}_3$, instead of running the SHVZK simulator, algorithm $\mathcal{P}$ is run to compute the commitment and the decommitment information. Note that the output of $\mathbf{H}_3$ corresponds to $(\mathtt{com}, T, \mathtt{dec}, m)$ where $(\omega, \mathtt{com}, T, \mathtt{rand}) \leftarrow\!\!\$\ \mathsf{Fake}_1^{\text{1-bit}}(\omega)$ and $\mathtt{dec} \leftarrow\!\!\$\ \mathsf{Fake}_2^{\text{1-bit}}(\omega, \mathtt{com}, m, \mathtt{rand})$. The hybrids $\mathbf{H}_2$ and $\mathbf{H}_3$ are perfectly indistinguishable due to the SHVZK property of Σ_{DH}.
- **Extractability.** If $T \notin L$, and $(\mathtt{com} := a, T)$ admits a valid opening for some message m, then there exists z such that $\mathcal{V}(\omega, T, a, m, z) = 1$. As discussed in §5.1, the latter implies that $\mathsf{ChallExt}$ extracts m.

□

5.2.2 Extending the Message Space

The extractable IDTC described above works for one-bit messages. It is easy to construct an extractable IDTC for messages of length k, for any $k \in \mathbb{N}$, by repeating an extractable IDTC for 1-bit messages k times. However, this means that the size of the CRS grows with the size of the messages. Moreover, for each bit of the message a different theorem would be sent. In this section, we show that the same CRS can be used by many instantiations of $\Pi^{\mathsf{1\text{-}bit}}$, and that the same tuple T can also be re-used in the commitment phase.

Fore sake of completeness, we describe the final scheme $\Pi^{\mathsf{k\text{-}bit}}$ formally.

- $\mathsf{Gen}^{\mathsf{k\text{-}bit}}$ runs the group generator GG (see §2.4) thus obtaining g, q, and outputs (ω, τ) with $\omega := (g, h)$ and $h = g^{\tau}$ for a random $\tau \in \mathbb{Z}_q$.
- $\mathsf{Com}^{\mathsf{k\text{-}bit}}$ takes as input $\omega = (g, h)$ and the message $m \in \{0,1\}^k$, picks $\alpha \leftarrow\!\!\$\, \mathbb{Z}_q$, computes a 1-non-DH tuple $T = (g, h, g^{\alpha}, gh^{\alpha})$, parses m as a k-bit string $m_1, \ldots, m_k$ and for each $i \in [k]$ lets $(a_i, z_i) \leftarrow\!\!\$\, \mathcal{S}(T, m_i)$. The output of $\mathsf{Com}^{\mathsf{k\text{-}bit}}$ corresponds to $\mathtt{com} := (a_i)_{i\in[k]}$, T, α and $\mathtt{dec} := (z_i)_{i\in[k]}$.
- $\mathsf{Dec}^{\mathsf{k\text{-}bit}}$ takes as input $\omega := (g, h)$, $\mathtt{com} := (a_i)_{i\in[k]}$, T, $\mathtt{dec} := (z_i)_{i\in[k]}$ and m, parses m as a k-bit string $m_1, \ldots, m_k$ and T as (g', h', X, W) and accepts if and only if $\omega = (g', h')$ and $\mathcal{V}(T, a_i, m_i, z_i) = 1$ for each $i \in [k]$.
- $\mathsf{Fake}_1^{\mathsf{k\text{-}bit}}$ takes as input $\omega = (g, h)$, picks $\alpha \leftarrow\!\!\$\, \mathbb{Z}_q$ and computes a DH tuple $T = (g, h, g^{\alpha}, h^{\alpha})$ and for each $i \leftarrow 1, \ldots, k$ computes $(a_i, st_i) \leftarrow\!\!\$\, \mathcal{P}(T, \alpha)$. The output of $\mathsf{Com}^{\mathsf{k\text{-}bit}}$ corresponds to $\mathtt{com} := (a_i)_{i\in[k]}$, $T := (g^{\alpha}, h^{\alpha})$, $\mathtt{rand} := (st_i)_{i\in[k]}$.
- $\mathsf{Fake}_2^{\mathsf{k\text{-}bit}}$ takes as input ω, the message $m \in \{0,1\}^k$, $\mathtt{com} := (a_i)_{i\in[k]}, \mathtt{rand} := (st_i)_{i\in[k]}$, parses m as a k-bit string $m_1, \ldots, m_k$ and for each $i \in [k]$ lets $z_i \leftarrow\!\!\$\, \mathcal{P}(st_i, m_i)$. The output of $\mathsf{Fake}_2^{\mathsf{k\text{-}bit}}$ corresponds to $(z_i)_{i\in[k]}$.
- $\mathsf{Ext}^{\mathsf{k\text{-}bit}}$ takes as input $(\omega, \mathtt{com} := (a_i)_{i\in[k]}, T, \tau)$, and for each $i \in [k]$ runs $\mathsf{ChallExt}(\omega, T, \mathtt{com}_i, \tau)$ thus obtaining m_i. The output of $\mathsf{Ext}^{\mathsf{k\text{-}bit}}$ corresponds to $m_1 || \ldots || m_k$.

The proof of the theorem below follows along the lines of the proof of Theorem 7 and is therefore omitted.

Theorem 8. *Assuming that the DDH assumption holds, then* $\Pi^{\mathsf{k\text{-}bit}}$ *is an extractable IDTC with message space* $\{0,1\}^k$.

5.2.3 Sigma Protocols for Extractable IDTCs

In this section we show how to construct a Sigma protocol for extractable IDTCs that allows proving that a commitment is computed in binding mode. To do that, we just need to provide a Sigma protocol that proves the instance T that accompanies the commitment $\mathtt{com}$ does not belong to L. Indeed, in this case the binding property of the extractable IDTC guarantees that $\mathtt{com}$ can be opened to only one value.

More formally, let $\Pi = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Dec}, \mathsf{Fake}_1, \mathsf{Fake}_2, \mathsf{Ext})$ be an extractable IDTC and ω be the output of Gen, we consider the following language:

$$L_{\Pi} = \big\{(\omega, \mathtt{com}, T) : \nexists \mathtt{dec}_0, m_0, \mathtt{dec}_1, m_1 \text{ with } m_0 \neq m_1 \text{ s.t.}$$
$$\mathsf{Dec}(\omega, \mathtt{com}, T, \mathtt{dec}_0, m_0) = 1 \text{ and } \mathsf{Dec}(\omega, \mathtt{com}, T, \mathtt{dec}_1, m_1) = 1\big\}.$$

The binding property of Π implies that in order to construct a Sigma protocol for the language L_{Π} associated to an extractable IDTC Π for language L, it suffices to construct a Sigma protocol for the language $\overline{L} = \{T : T \notin L\}$. Hence, for the concrete case of $\Pi^{\mathsf{k\text{-}bit}}$, we can just use a Sigma protocol Σ_{1nDH} for the relation R_{1nDH} (see §2.4).

Common input: Security parameter λ, and instance $x \in L$ (available only in the last round).
Prover's private input: Witness w (available only in the last round) s.t. $(x, w) \in R$.
CRS generation: Run $(\tilde{\omega}, \tilde{\tau}) \leftarrow\!\!\$\ \mathsf{GenIDTC}(1^\lambda)$ and $(\omega, \tau) \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$. Output $\omega'' := (\omega, \tilde{\omega})$.
$\mathcal{P}''(1^\lambda, 1^n)$**:** The first prover's message is generated as follows.

- Compute $(a, st) \leftarrow\!\!\$\ \mathcal{P}(1^\lambda, 1^n)$.
- Run Com on input ω and a, thus obtaining $(\mathtt{com}, T, \alpha, \mathtt{dec})$.
- Compute $(a', st') \leftarrow\!\!\$\ \mathcal{P}'(T, \alpha)$.
- Send $a'' := (\mathtt{com}, T, a')$ to $\mathcal{V}''$ and set $st'' = (st, st', \mathtt{dec}, a)$.

$\mathcal{V}''(1^\lambda)$**:** Pick $c \leftarrow\!\!\$\ \{0,1\}^\ell$ and send it to $\mathcal{P}''$.
$\mathcal{P}''(x, w, st'', c)$**:** The last prover's message is generated as follows.

- Compute $z \leftarrow\!\!\$\ \mathcal{P}(x, w, st, c)$.
- Compute $z' \leftarrow\!\!\$\ \mathcal{P}'(st', c)$.
- Send $z'' := (z, z', (\mathtt{dec}, a))$ to $\mathcal{V}''$.

Verifier's verdict: Accept if and only if the following conditions are satisfied: (i) $\mathsf{Dec}(\tilde{\omega}, \mathtt{com}, T, \mathtt{dec}, a) = 1$; (ii) $\mathcal{V}(\omega, x, a, c, z) = 1$; (iii) $\mathcal{V}'(T, a', c, z') = 1$;

Figure 4: Adding adaptive-input SHVZK to delayed-input trapdoor Sigma protocols.

5.3 Trapdoor Sigma Protocols with Adaptive SHVZK

We are finally ready to show how our last compiler works. Our transform can be seen as a modified version of the compiler proposed in §3, with the difference that we now start with a delayed-input trapdoor Sigma protocol Σ with (not necessarily adaptive-input) SHVZK and the compiler outputs a delayed-input trapdoor Sigma protocol Σ'' that enjoys adaptive-input SHVZK. Then, by using Theorem 5, we can obtain a non-interactive argument system satisfying adaptive zero knowledge in the CRS model. We make use of the following ingredients:

- A delayed-input trapdoor Sigma protocol $\Sigma = (\mathsf{Gen}, \mathcal{P}, \mathcal{V}, \mathsf{TrapGen}, \mathsf{BadChallenge})$.
- The extractable IDTC $\Pi = (\mathsf{GenIDTC}, \mathsf{Com}, \mathsf{Dec}, \mathsf{Fake}_1, \mathsf{Fake}_2, \mathsf{Ext})$ for the relation R_{DH} described in §5.2.2.
- The Sigma protocol $\Sigma' = (\mathcal{P}', \mathcal{V}')$ for the relation R_{1nDH} described in §5.1. We recall that such a Sigma protocol admits a bad-challenge extractor algorithm $\mathsf{ChallExt}$.

The core idea of our protocol is exactly the one proposed in §3. The main difference is that we use an extractable IDTC instead of an IDTC. The reason for this change is that we need to prove that our protocol admits an algorithm $\mathsf{BadChallenge}''$. We recall that $\mathsf{BadChallenge}''$ takes as input a false statement x, the first round of the protocol and a trapdoor, and outputs the bad challenge. The first round of our protocol is represented by a commitment (computed using the extractable IDTC Π) to the first round of Σ. Hence, we can rely on the extractability property of Π in order to extract the first round of Σ and finally run the algorithm $\mathsf{BadChallenge}$ associated to Σ (which exists by definition) to complete the extraction of the bad challenge.

This approach works under the assumption that the commitment computed in the first round is binding. Indeed, the extractor of Π guarantees extraction of the correct value only when the commitment is computed with respect to a false instance. This is where we rely on Σ'. Indeed, in this case the statement proved under Σ' is false, and thus we can extract the bad challenge from Σ' by running $\mathsf{ChallExt}$. A formal description of Σ'' appears in Fig. 4.

Theorem 9. *Let R be a polynomial-time relation. Assume that Σ is a delayed-input trapdoor Sigma protocol for R with standard SHVZK, that Π is the extractable IDTC for R_{DH} described in §5.2.2, and that Σ' is the Sigma protocol for the relation R_{1nDH} described in §5.1. Then, the Sigma protocol Σ'' of Fig. 4 is a delayed-input trapdoor Sigma protocol for R satisfying adaptive-input SHVZK under the DDH assumption.*

Simulation of the CRS. Run $\mathsf{GenIDTC}(1^\lambda)$ thus obtaining $(\tilde{\omega}, \tilde{\tau})$, and $\mathsf{Gen}(1^\lambda)$ thus obtaining (ω, τ). Set the CRS as $\omega'' := (\omega, \tilde{\omega})$.

Simulation of the first round: Upon input 1^λ, ω'' and $c \in \{0,1\}^\lambda$, do the following.

- Compute $(\mathtt{com}, T, \mathtt{rand}) \leftarrow\!\!\$\ \mathsf{Fake}_1(\tilde{\omega})$.
- Compute $(a', z') \leftarrow\!\!\$\ \mathcal{S}'(T, c)$.
- Output $a := (\mathtt{com}, T, a')$.

Simulation of the last round: Upon input ω'' and $x \in L$, do the following.

- Compute $(a, z) \leftarrow\!\!\$\ \mathcal{S}(\omega, x, c)$.
- Compute $\mathtt{dec} \leftarrow\!\!\$\ \mathsf{Fake}_2(\tilde{\omega}, \mathtt{com}, a, \mathtt{rand})$
- Output $z'' := (z, z', (\mathtt{dec}, a))$.

Figure 5: Adaptive SHVZK simulator.

Proof. We start the proof by showing that Σ'' is a trapdoor Sigma protocol, and then we show that it is also adaptive-input SHVZK. The algorithms $\mathsf{TrapGen}''$ and $\mathsf{BadChallenge}''$ are described below.

- Algorithm $\mathsf{TrapGen}''$ runs $\mathsf{TrapGen}(1^\lambda)$ thus obtaining (ω, τ) and $\mathsf{GenIDTC}(1^\lambda)$ thus obtaining $(\tilde{\omega}, \tilde{\tau})$, and sets $\omega'' := (\omega, \tilde{\omega})$ and $\tau'' := (\tau, \tilde{\tau})$. The CRS indistinguishability property follows directly from the CRS indistinguishability of $\mathsf{TrapGen}$.
- Algorithm $\mathsf{BadChallenge}''$ works as follows. Upon input τ'', ω'', x, a'', it first parses a'' as $(\mathtt{com}, T, a')$, ω'' as $(\omega, \tilde{\omega})$ and τ'' as $(\tau, \tilde{\tau})$. Hence:
  - If T is a non-DH tuple (note that given $\tilde{\tau}$ it is efficient to check whether a tuple T is DH or not), then it runs $\mathsf{Ext}(\tilde{\omega}, T, \mathtt{com}, \tilde{\tau})$ thus obtaining a and outputs $\mathsf{BadChallenge}(\tau, \omega, x, a)$.
  - Else, it outputs $\mathsf{ChallExt}(T, a', \tilde{\tau})$ and stops.[18]

We now argue that if $x \notin L$ then the output of $\mathsf{BadChallenge}''$ is correct in the following cases (which capture all possible scenarios): (i) T is a non-DH tuple; (ii) T is a DH tuple. Suppose that we are in case (i). Then the extraction property of the extractable IDTC guarantees that the messages committed in $(\mathtt{com}, T)$ can be extracted with probability 1 (note that in this case all the commitments are perfectly binding). Hence, we can rely on $\mathsf{BadChallenge}$ to extract the bad challenge from a, which is the only possible value to which $(\mathtt{com}, T)$ can be opened to.

On the other hand, suppose that we are in case (ii). In this case, we cannot rely on the extractability property of Π since $T \notin L_{1nDH}$. However, we know that the statement proved under Σ' is false, and we can thus run $\mathsf{ChallExt}(T, a', \tilde{\tau})$ in order to obtain the only challenge c such that $\mathcal{V}'(T, a', c, z') = 1$. This proves correctness.

It remains to show that Σ'' satisfies adaptive-input SHVZK. The proof of this fact is very similar to the proof of Lemma 3, and thus we only give a brief sketch here. Let $\mathcal{S}$ be the SHVZK simulator of Σ, and $\mathcal{S}'$ be the SHVZK of Σ'. The adaptive-input SHVZK simulator of Σ'' is described in Fig. 5. To argue indistinguishability, we use a hybrid argument. The first hybrid experiment $\mathbf{H}_1$ corresponds to the game in which the honest prover procedure is executed to compute a proof. In $\mathbf{H}_2$, upon receiving c from the adversary, the SHVZK simulator of Σ' is run in order to compute a', z'. That is, $\mathbf{H}_2$ runs $(a', z') \leftarrow\!\!\$\ \mathcal{S}'(T, c)$ and uses a', z' to complete the transcript of Σ'. In $\mathbf{H}_3$, the commitment $(\mathtt{com}, T)$ is computed by running Fake_1, whereas the decommitment information $\mathtt{dec}$ is obtained by running Fake_2. That is, $\mathbf{H}_3$ runs $(\mathtt{com}, T, \mathtt{rand}) \leftarrow\!\!\$\ \mathsf{Fake}_1(\tilde{\omega})$ and $\mathtt{dec} \leftarrow\!\!\$\ \mathsf{Fake}_2(\tilde{\omega}, \mathtt{com}, a, \mathtt{rand})$. Finally, in $\mathbf{H}_4$ the SHVZK of Σ is run in order to compute (a, z). That is, $\mathbf{H}_4$ runs $(a, z) \leftarrow\!\!\$\ \mathcal{S}(\omega, x, c)$. □

[18] We recall that, accordingly to what we have described in §5.2.2, $\tilde{\omega} := (g, g^\gamma)$ and $\tilde{\tau} = \gamma$. Given that $T = (g, g^\gamma, A, B)$ for some A, B, we can indeed run $\mathsf{ChallExt}$ on input the tuple T, the first round a' of Σ', and γ.

6 Conclusions

We have studied adaptive security of delayed-input Sigma protocols and Fiat-Shamir NIZKs in the CRS model derived from trapdoor Sigma protocols using CI hash functions. In particular, one of our results clarifies that hashing both the prover's first round and the instance being proven in the Fiat-Shamir transform yields NIZKs with adaptive soundness (but not necessarily adaptive zero knowledge), whereas hashing only the prover's first round yields adaptive[19] zero knowledge (but not necessarily adaptive soundness). This contradicts the seemingly folklore belief that Fiat-Shamir NIZKs obtained from trapdoor Sigma protocols following the paradigm of Canetti *et al.* [CCH+19] achieve adaptive security. (For instance, the latter is stated explicitly in [BKM20, Theorem 3.4] and erroneously attributed to [CCH+19].)

Interesting open problems include obtaining efficient delayed-input Sigma protocols and trapdoor Sigma protocols without computational assumptions, and understanding which flavor of the Fiat-Shamir transform for generic Sigma protocols allows to obtain NIZKs satisfying both adaptive soundness and adaptive zero knowledge in the CRS model (without NP reductions).

7 Acknowledgments

Michele Ciampi was supported by H2020 project PRIVILEDGE #780477.

[19] The latter additionally requires the starting Sigma protocol to satisfy adaptive-input SHVZK.

References

[ACJ17] Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 468–499, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 483–501, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd Annual Symposium on Foundations of Computer Science*, pages 106–115, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

[BDG+13] Nir Bitansky, Dana Dachman-Soled, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, Adriana López-Alt, and Daniel Wichs. Why "Fiat-Shamir for proofs" lacks a proof. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 182–201, Tokyo, Japan, March 3–6, 2013. Springer, Heidelberg, Germany.

[BG93] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Heidelberg, Germany.

[BGJ+18] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal MPC. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 459–487, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 645–677, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany.

[BKM20] Zvika Brakerski, Venkata Koppula, and Tamer Mour. NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. *IACR Cryptology ePrint Archive*, 2020:258, 2020.

[Blu86] Manuel Blum. How to prove a theorem so no one else can claim it. In *In Proceedings of the International Congress of Mathematicians*, page 444–451, 1986.

[BLV03] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. In *44th Annual Symposium on Foundations of Computer Science*,

pages 384–393, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press.

[BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 626–643, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.

[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: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[BR14] Mihir Bellare and Todor Ristov. A characterization of chameleon hash functions and new, efficient designs. *Journal of Cryptology*, 27(4):799–823, October 2014.

[CCG+19] Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. Cryptology ePrint Archive, Report 2019/216, 2019. https://eprint.iacr.org/2019/216.

[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 Annual ACM Symposium on Theory of Computing*, pages 1082–1090, Phoenix, AZ, USA, June 23–26, 2019. ACM Press.

[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: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 389–415, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[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, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 91–122, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, Santa Barbara, CA, USA, August 21–25, 1994. Springer, Heidelberg, Germany.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218, Dallas, TX, USA, May 23–26, 1998. ACM Press.

[COSV16] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Concurrent non-malleable commitments (and more) in 3 rounds. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume

9816 of *Lecture Notes in Computer Science*, pages 270–299, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[COSV17a] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Four-round concurrent non-malleable commitments from one-way functions. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 127–157, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[COSV17b] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Round-optimal secure two-party computation from trapdoor permutations. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 678–710, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany.

[CPS+16] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Online/offline OR composition of sigma protocols. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 63–92, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[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: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 83–111, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[CV05] Dario Catalano and Ivan Visconti. Hybrid trapdoor commitments and their applications. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005: 32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 298–310, Lisbon, Portugal, July 11–15, 2005. Springer, Heidelberg, Germany.

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

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

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 581–596, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany.

[DNRS99] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. In *40th Annual Symposium on Foundations of Computer Science*, pages 523–534, New York, NY, USA, October 17–19, 1999. IEEE Computer Society Press.

[FF02] Marc Fischlin and Roger Fischlin. The representation problem based on factoring. In Bart Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 96–113, San Jose, CA, USA, February 18–22, 2002. Springer, Heidelberg, Germany.

[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, *Progress in Cryptology - INDOCRYPT 2012: 13th International Conference in Cryptology in India*, volume 7668 of *Lecture Notes in Computer Science*, pages 60–79, Kolkata, India, December 9–12, 2012. Springer, Heidelberg, Germany.

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

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th Annual Symposium on Foundations of Computer Science*, pages 102–115, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press.

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

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GR19] Vipul Goyal and Silas Richelson. Non-malleable commitments using Goldreich-Levin list decoding. In David Zuckerman, editor, *60th Annual Symposium on Foundations of Computer Science*, pages 686–699, Baltimore, MD, USA, November 9–12, 2019. IEEE Computer Society Press.

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

[HKR+14] Feng Hao, Matthew Nicolas Kreeger, Brian Randell, Dylan Clarke, Siamak Fayyaz Shahandashti, and Peter Hyun-Jeen Lee. Every vote counts: Ensuring integrity in large-scale electronic voting. *USENIX Journal of Election Technology and Systems*, 2(3), July 2014.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. Information Security and Cryptography. Springer, 2010.

[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 Annual Symposium on Foundations of Computer Science*, pages 850–858, Paris, France, October 7–9, 2018. IEEE Computer Society Press.

[HMR08] Shai Halevi, Steven Myers, and Charles Rackoff. On seed-incompressible functions. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 19–36, San Francisco, CA, USA, March 19–21, 2008. Springer, Heidelberg, Germany.

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

[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, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 224–251, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[Lin15] Yehuda Lindell. An efficient transform from sigma protocols to NIZK with a CRS and non-programmable random oracle. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 93–109, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[LS91] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 353–365, Santa Barbara, CA, USA, August 11–15, 1991. Springer, Heidelberg, Germany.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany.

[MV16] Arno Mittelbach and Daniele Venturi. Fiat-Shamir for highly sound protocols is instantiable. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16: 10th International Conference on Security in Communication Networks*, volume 9841 of *Lecture Notes in Computer Science*, pages 198–215, Amalfi, Italy, August 31 – September 2, 2016. Springer, Heidelberg, Germany.

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Heidelberg, Germany.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398, Saragossa, Spain, May 12–16, 1996. Springer, Heidelberg, Germany.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 89–114, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.

[Ste94] Jacques Stern. A new identification scheme based on syndrome decoding. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 13–21, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Heidelberg, Germany.

$\underline{\mathbf{G}(\lambda, c, 1)}$:
$(T, \alpha) \leftarrow\!\!\$\ R_{DH}$

$(\mathtt{com}, \mathtt{rand}) \leftarrow\!\!\$\ \mathsf{Fake}_1(T, \alpha)$
$(a', z') \leftarrow\!\!\$\ \mathcal{S}'(T, c)$
$a'' = (\mathtt{com}, a', T)$
$(x, w) \leftarrow\!\!\$\ \mathcal{A}(a'', c)$
$(a, z) \leftarrow\!\!\$\ \mathcal{S}(x, c)$
$\mathtt{dec} \leftarrow\!\!\$\ \mathsf{Fake}_2((T, \alpha), a, (\mathtt{com}, \mathtt{rand}))$
If $(x, w) \in R$
  $z'' = (z, z', (\mathtt{dec}, a))$
Else $z'' = \bot$
$b' \leftarrow\!\!\$\ \mathcal{A}((x, w), (a'', c, z''))$

$\underline{\mathbf{H}_1(\lambda, c)}$:
$(T, \alpha) \leftarrow\!\!\$\ R_{DH}$
$(a, st) \leftarrow\!\!\$\ \mathcal{P}(1^\lambda)$
$(\mathtt{com}, \mathtt{rand}) \leftarrow\!\!\$\ \mathsf{Fake}_1(T, \alpha)$
$(a', z') \leftarrow\!\!\$\ \mathcal{S}'(T, c)$
$a'' = (\mathtt{com}, a', T)$
$(x, w) \leftarrow\!\!\$\ \mathcal{A}(a'', c)$
$z \leftarrow\!\!\$\ \mathcal{P}(x, w, a, c)$
$\mathtt{dec} \leftarrow\!\!\$\ \mathsf{Fake}_2((T, \alpha), a, (\mathtt{com}, \mathtt{rand}))$
If $(x, w) \in R$
  $z'' = (z, z', (\mathtt{dec}, a))$
Else $z'' = \bot$
$b' \leftarrow\!\!\$\ \mathcal{A}(x, (a'', c, z''))$

$\underline{\mathbf{H}_2(\lambda, c)}$:
$(T, \alpha) \leftarrow\!\!\$\ R_{1nDH}$
$(a, st) \leftarrow\!\!\$\ \mathcal{P}(1^\lambda)$
$(\mathtt{com}, \mathtt{dec}) \leftarrow\!\!\$\ \mathsf{Com}(T, a)$
$(a', z') \leftarrow\!\!\$\ \mathcal{S}'(T, c)$
$a'' = (\mathtt{com}, a', T)$
$(x, w) \leftarrow\!\!\$\ \mathcal{A}(a'', c)$
$z \leftarrow\!\!\$\ \mathcal{P}(x, w, c, st)$
If $(x, w) \in R$
  $z'' = (z, z', (\mathtt{dec}, a))$
Else $z'' = \bot$
$b' \leftarrow\!\!\$\ \mathcal{A}(x, (a'', c, z''))$

$\underline{\mathbf{G}(\lambda, c, 0)}$:
$(T, \alpha) \leftarrow\!\!\$\ R_{1nDH}$
$(a, st) \leftarrow\!\!\$\ \mathcal{P}(1^\lambda)$
$(\mathtt{com}, \mathtt{dec}) \leftarrow\!\!\$\ \mathsf{Com}(T, a)$
$(a', st') \leftarrow\!\!\$\ \mathcal{P}'(T, \alpha)$
$a'' = (\mathtt{com}, a', T)$
$(x, w) \leftarrow\!\!\$\ \mathcal{A}(a'', c)$
$z \leftarrow\!\!\$\ \mathcal{P}(x, w, a, c)$
$z' \leftarrow\!\!\$\ \mathcal{P}'(st', c)$
If $(x, w) \in R$
  $z'' = (z, z', (\mathtt{dec}, a))$
Else $z'' = \bot$
$b' \leftarrow\!\!\$\ \mathcal{A}(x, (a'', c, z''))$

Figure 6: Hybrid experiments in the proof of Theorem 3