Dual-System Simulation-Soundness with Applications to UC-PAKE and More

Charanjit S. Jutla
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
csjutla@us.ibm.com

Arnab Roy
Fujitsu Laboratories of America
Sunnyvale, CA 94085, USA
arnab@cs.stanford.edu

Abstract

We introduce a novel concept of dual-system simulation-sound non-interactive zero-knowledge (NIZK) proofs. Dual-system NIZK proof system can be seen as a two-tier proof system. As opposed to the usual notion of zero-knowledge proofs, dual-system defines an intermediate partial-simulation world, where the proof simulator may have access to additional auxiliary information about the potential language member, for example a membership bit, and simulation of proofs is only guaranteed if the membership bit is correct. Further, dual-system NIZK proofs allow a quasi-adaptive setting where the CRS can be generated based on language parameters. This allows for the further possibility that the partial-world CRS simulator may have access to additional trapdoors related to the language parameters. We show that for important hard languages like the Diffie-Hellman language, such dual-system proof systems can be given which allow unbounded partial simulation soundness, and which further allow transition between partial simulation world and single-theorem full simulation world even when proofs are sought on non-members. The construction is surprisingly simple, involving only two additional group elements for general linear-subspace languages in asymmetric bilinear pairing groups.

As a direct application we give a short keyed-homomorphic CCA-secure encryption scheme. The ciphertext in this scheme consists of only six group elements (under the SXDH assumption) and the security reduction is tight. An earlier scheme of Libert et al based on their efficient unbounded simulation-sound QA-NIZK proofs only provided a loose security reduction, and further had ciphertexts almost twice as long as ours.

We also show a single-round universally-composable password authenticated key-exchange (UC-PAKE) protocol which is secure under adaptive corruption in the erasure model. The single message flow only requires **four** group elements under the SXDH assumption. This is the *shortest known* UC-PAKE even without considering adaptive corruption. The latest published scheme which considered adaptive corruption, by Abdalla et al [ABB$^+$13], required non-constant (more than 10 times the bit-size of the password) number of group elements. For fully adaptive corruption, we realize a relaxed ideal functionality that uses non-information oracles.

Keywords: NIZK, bilinear pairings, UC-PAKE, keyed-homomorphic encryption, SXDH.

Contents

1 Introduction

Since the introduction of simulation-sound non-interactive zero-knowledge (NIZK) proofs in [Sah99] (based on the concept of non-malleability [DDN91]), simulation-soundness has become an essential cryptographic tool. While the idea of zero-knowledge simulation [GMR89] brought rigor to the concept of semantic security, simulation-soundness of some form is usually implicit in most cryptographic applications. While the original construction of [Sah99] was rather inefficient, the advent of pairing based cryptography, and in particular Groth-Sahai NIZK proofs [GS08], has led to much more efficient simulation-sound NIZK constructions. Pairing-based cryptography has also led to efficient construction of powerful primitives where simulation-soundness is not very explicit.

It has been shown that different forms of simulation-soundness suffice for many applications. Indeed, the original application (CCA2-secure encryption) considered in [Sah99] only required what is known as single-theorem simulation-soundness (also known as one-time simulation-soundness). However, many other cryptographic constructions are known only using unbounded simulation-sound NIZK proofs. In this paper, we introduce the concept of **dual-system simulation-sound** NIZK proofs, which lie somewhere in between one-time and unbounded simulation-sound NIZK proofs. The aim is to show that this weaker concept suffices for constructions where unbounded simulation-soundness was being used till now. We also show that in many applications this new concept of dual-system simulation soundness is implicit, in the sense that although we cannot get a generic construction from a NIZK proof, we can use the underlying ideas of the dual-system simulation-sound NIZK proofs.

Indeed, our novel definition is inspired by the dual-system identity-based encryption (IBE) scheme of Waters [Wat09], where such a concept was implicit, and led to the first IBE scheme which was fully-secure under static and standard assumptions. So without further ado, we jump straight into the main idea of the new concept. In dual-system simulation-sound NIZK proof systems we will consider three worlds: the real-world, the partial-simulation world, and the one-time full-simulation world. The real world consists of a common-reference string (CRS), an efficient prover P, and an efficient verifier V. The concept of completeness and soundness of P and V with respect to a witness-relation R is well-understood. The full-simulation world is also standard, and it includes two simulators: a CRS simulator and a proof simulator. The proof simulator is a zero-knowledge simulator in the sense that it can simulate proofs even without access to the witness. In order to achieve this, the CRS simulator generates the CRS in a potentially different way and produces a trapdoor for the proof simulator. The **partial-simulation** world we consider also has a CRS simulator, and a proof simulator, but this proof simulator is allowed partial access to the witness (or some other auxiliary information) about the member on which the proof is sought.

At this point, we also bring in the possibility of the CRS being generated as a function of the language or witness-relation under consideration. The recent quasi-adaptive NIZK (QA-NIZK) proofs of [JR13] allow this possibility for distributions of witness-relations. The CRS in the real and the full-simulation world is generated based on a language parameter generated according to some distribution. Now we consider the possibility that in the partial-simulation world, the CRS simulator actually generates the language parameter itself. In other words, the CRS simulator has access to the "witness" of the language parameter. For example, the CRS simulator may know the discrete-logs of the language parameters. This leads to the possibility that in the partial simulation world the proof simulator may have access to additional trapdoors which makes simulation and/or simulation soundness easier to achieve.

In this paper, we will only define and consider dual-system simulation sound QA-NIZK proofs (called DSS-QA-NIZK), where the only auxiliary information that the partial proof simulator gets is a single bit which is called the **membership bit**. The membership bit indicates whether the statement on which the proof is sought is in the language or not. We show that we can achieve unbounded partial-simulation soundness for important languages like the Diffie-Hellman language by a relatively simple constructions. The constructions also allow one-time full-ZK simulation, and hence form a DSS-QA-NIZK for the Diffie-Hellman language. We actually give a general construction for arbitrary languages which allow smooth and universal$_2$ projective hash proofs [CS02], and such that the language augmented with such a hash proof has a usual QA-NIZK. We show that for linear subspace languages (over bilinear groups), like the Diffie-Hellman and decisional-linear (DLIN) languages, the requirements for the general construction are easy to obtain. Thus, for all such languages, under the standard and static SXDH assumption in bilinear pairing groups, we get a DSS-QA-NIZK proof of only two group elements.

Table 1 summarizes comparison among existing schemes and ours. DSS is weaker than unbounded simulation soundness, and although incomparable with one time simulation soundness, it seems to enjoy better properties. Consistent with this, we observe that the proof sizes also place in the middle of the shortest known OTSS-NIZKs [ABP15, KW15] and the shortest known USS-NIZKs [KW15] for linear subspaces.

Table 1: Comparison with existing NIZK schemes for linear subspaces with table adapted from [KW15]. The language of interest is a t dimensional subspace of an n dimensional ambient space. m is the bit-size of the tag. AS is adaptive-soundness. OTSS is one-time simulation-soundness and USS is unbounded simulation-soundness.

	Soundness	Assumption	Proof	CRS	#pairings
[GS08]	AS	DLIN	$2n+3t$	6	$3n(t+3)$
[LPJY14]	AS	DLIN	3	$2n+3t+3$	$2n+4$
[JR13]	AS	k-Linear	$k(n-t)$	$2kt(n-t)+k+1$	$k(n-t)(t+2)$
[JR14]	AS	k-Linear	k	$kn+kt+k^2$	$kn+k^2$
[ABP15]	AS	k-Linear	k	$kn+kt+k$	$kn+k$
[KW15]	AS	k-Linear	k	$kn+kt+k-1$	$kn+k-1$
[ABP15]	OTSS	k-Linear	k	$2m(kn+(k+1)t)+k$	$mkn+k$
[KW15]	OTSS	k-Linear	k	$2m(kn+(k+1)t)$+k-1	mkn+k-1
This paper	DSS	k-Linear	$k+1$	$k(n+1)+kt+k^2$	$k(n+1)+k^2$
[CCS09]	USS	DLIN	$2n+6t+52$	18	$O(tn)$
[LPJY14]	USS	DLIN	20	$2n+3t+3m+10$	$2n+30$
[KW15]	USS	k-Linear	$2k+2$	$kn+4(k+t+1)k+2k$	$k(n+k+1)+k$

1.0.1 Applications.

We now give the main idea as to why such a construction is useful. The security of most applications is shown by reduction to a hard language. However, a particular application may have a more complex language for which the NIZK proofs are required, and the security proof may require soundness of the NIZK system while proofs of many elements (real or fake) of such a complex language are being simulated. The idea is that multiple simulations of such elements can be performed in a partial-simulation manner (i.e. it is always possible to supply the correct membership-bit), and full simulation is only required of one member at a time, on which the hardness assumption can then be invoked.

Keyed-Homomorphic CCA-secure Encryption. As a first application we consider the keyed-homomorphic CCA-secure encryption scheme notion of [EHO$^+$13]. In such an encryption scheme, a further functionality called Eval is available which using a key can homomorphically combine valid ciphertexts. The scheme should provide IND-CCA2 security when this Eval key is unavailable to the adversary, and should continue to enjoy IND-CCA1 security when the Eval key is exposed to the adversary. Emura et al. also gave constructions for such a scheme, albeit schemes which are not publicly verifiable, and further satisfying a weaker notion than CCA1-security when Eval key is revealed. Recently, Libert et al gave a publicly-verifiable construction which is more efficient and also CCA1-secure when Eval key is revealed. Their construction is based on a new and improved unbounded simulation-sound QA-NIZK for linear subspace languages. We show in this paper that a DSS-QA-NIZK for the Diffie-Hellman language suffice, and leads to a much improved construction. While the construction in [LPJY14], under the SXDH assumption, requires nine group elements in one group, and two more in the other plus a one-time signature key pair, our construction *only requires six group elements* in any one of the bilinear groups. Further, while the earlier construction was loose (i.e. looses a factor quadratic in number of Eval calls), our reduction is tight.

Universally-Composable Password-Authenticated Key Exchange (UC-PAKE). The UC-PAKE ideal functionality $\mathcal{F}_{\text{PAKE}}$ was introduced in [CHK$^+$05] where they also gave a three-round construction. In [KV11] a single-round construction for UC-PAKE was given using Groth-Sahai NIZK proofs along with unbounded simulation-soundness construction of [CCS09] (also see [JR12]). Later [BBC$^+$13] gave a UC-PAKE construction based on novel trapdoor smooth projective hash functions, but secure only under static corruption; each message consisted of six group elements in one group, and another five elements in the other group (under the SXDH assumption).

In this paper, we construct a a single-round construction based on dual-system simulation-soundness which is UC-secure under adaptive corruption (in the erasure model), and which has only a total of **four** group elements in each message. The key is generated in the target group. The construction is not a black-box application of the DSS-QA-NIZK for the Diffie-Hellman language, but uses its underlying idea as well as the various component algorithms of the DSS-QA-NIZK. The main idea of the construction is given in more detail in Section 6.3.

To the best of our knowledge, this is the *shortest known UC-PAKE*, even without considering adaptive corruption. The first UC-PAKE to consider adaptive corruption was by Abdalla, Chevalier and Pointcheval [ACP09], which was a two round construction. Recently, Abdalla et al [ABB$^+$13] also constructed a single round protocol, which required a non-constant (more than 10 times the bit-size of the password) number of group elements in each flow. Comparison with existing UC-PAKEs is given in Table 2.

Our result only obtains adaptive security in synchronous models, where either one party sends the first message in phase one and the second party responds in phase two (all phases being time bound), or where both parties simulataneously send a message to the network adversary in round one and adversary deivers both messages simultaneously in round two. For adaptive corruption in the asynchronous model, where the adversary can corrupt a party even after the peer has emitted its session key, we can only realize a relaxed ideal functionality $\mathcal{F}_{\text{RPAKE}}$ that uses a *non-information oracle* [CK02]. This is really a technical restriction[1], as it can be shown that UC-secure (password-based) constant message-length secure channel, for any constant, can be realized in the $\mathcal{F}_{\text{RPAKE}}$-hybrid model [CK02] by a simple one-message protocol. Variable-length secure channel

[1] More details as well as a discussion about the relaxed functionality can be found in Section 6.2.

realization anyway requires unnatural protocols as it is impossible to realize non-interactive non-committing encryption in the standard model [Nie02]. Alternatively, one can consider a relaxed non-information oracle based variable message-length secure channel, which can then be realized using $\mathcal{F}_{\text{RPAKE}}$ [CK02].

Table 2: Comparison with existing UC-PAKE schemes. m is the password size in bits and λ is the security parameter. AC stands for Adaptive Corruption. For one-round schemes, message size is per flow.

	AC	One-round	Assumption	Message size
[ACP09]	yes	no	DDH	$O(m\lambda)$
[KV11]	no	yes	DLIN	$> 65 \times \mathbb{G}$
[JR12]	no	yes	SXDH	> 30 total group elements
[BBC$^+$13]	no	yes	SXDH	$6 \times \mathbb{G}_1 + 5 \times \mathbb{G}_2$
[ABB$^+$13]	yes	yes	SXDH	$10 * m \times \mathbb{G}_1 + m \times \mathbb{G}_2$
This paper	yes	yes	SXDH	$3 \times \mathbb{G}_1 + 1 \times \mathbb{G}_2$

In Appendix F, we show that the recent efficient dual-system IBE [JR13] (inspired by the original dual-system IBE of Waters [Wat09]) can also be obtained using the ideas of DSS-QA-NIZK. While the construction is not black-box and utilizes additional "smoothness" and "single-pairing-product test" properties of the verifier, it along with the other two applications clearly demonstrate the power and utility of the new notion, which we expect will find many more applications.

2 Preliminaries: Quasi-Adaptive NIZK Proofs

A witness relation is a binary relation on pairs of inputs, the first called a (potential) language member and the second called a witness. Note that each witness relation R defines a corresponding language L which is the set of all x for which there exists a witness w, such that $R(x, w)$ holds.

We will consider Quasi-Adaptive NIZK proofs [JR13] for a probability distribution $\mathcal{D}$ on a collection of (witness-) relations $\mathcal{R} = \{R_\rho\}$ (with corresponding languages L_ρ). Recall that in a quasi-adaptive NIZK, the CRS can be set after the language parameter has been chosen according to $\mathcal{D}$. Please refer to [JR13] for detailed definitions.

Definition 1 *([JR13]) We call* $(\mathsf{pargen}, \mathsf{crsgen}, \mathsf{prover}, \mathsf{ver})$ *a* (labeled) quasi-adaptive non-interactive zero-knowledge *(QA-NIZK) proof system for witness-relations* $\mathcal{R}_\lambda = \{R_\rho\}$ *with parameters sampled from a distribution* $\mathcal{D}$ *over associated parameter language* Lpar*, if there exist such that for all non-uniform PPT adversaries* $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ *we have (in all of the following probabilistic experiments, the experiment starts by setting* λ *as* $\lambda \leftarrow \mathsf{pargen}(1^m)$*, and choosing* ρ *as* $\rho \leftarrow \mathcal{D}_\lambda$*):*

Quasi-Adaptive Completeness:
$\Pr[\text{CRS}\leftarrow \mathsf{crsgen}(\lambda, \rho); (x, w, l) \leftarrow \mathcal{A}_1(\text{CRS}, \rho);$
$\pi \leftarrow \mathsf{prover}(\text{CRS}, x, w, l) : \ \mathsf{ver}(\text{CRS}, x, l, \pi) = 1 \ \textit{if } R_\rho(x, w)] = 1$

Quasi-Adaptive Soundness:
$\Pr[\text{CRS}\leftarrow \mathsf{crsgen}(\lambda, \rho); (x, l, \pi) \leftarrow \mathcal{A}_2(\text{CRS}, \rho) : x \notin L_\rho \ \wedge \ \mathsf{ver}(\text{CRS}, x, l, \pi) = 1] \approx 0$

Quasi-Adaptive Zero-Knowledge:
$\Pr[\textsc{crs} \leftarrow \mathsf{crsgen}(\lambda, \rho) : \mathcal{A}_3^{\mathsf{prover}(\textsc{crs},\cdot,\cdot,\cdot)}(\textsc{crs}, \rho) = 1] \approx$
$\Pr[(\textsc{crs}, \mathsf{trap}) \leftarrow \mathsf{crs\text{-}sim}(\lambda, \rho) : \mathcal{A}_3^{\mathsf{sim}^*(\textsc{crs},\mathsf{trap},\cdot,\cdot,\cdot)}(\textsc{crs}, \rho) = 1]$,
where $\mathsf{sim}^*(\textsc{crs}, \mathsf{trap}, x, w, l) = \mathsf{sim}(\textsc{crs}, \mathsf{trap}, x, l)$ *for* $(x, w) \in R_\rho$ *and both oracles (i.e.* prover *and* sim^**) output failure if* $(x, w) \notin R_\rho$.

The QA-NIZK is called a **statistical zero-knowledge** QA-NIZK if the view of adversary $\mathcal{A}_3$ above in the two experiments is statistically indistinguishable.

3 Dual-System Simulation-Soundness

To define dual-system simulation soundness of QA-NIZK proofs, we will consider three worlds: the real-world, the partial-simulation world, and the one-time (or single theorem) full-simulation world. While the real-world and the full-simulation world should be familiar from earlier definitions of NIZK proof systems, the partial-simulation world leads to interesting possibilities. To start with, in the partial simulation world, one would like the proof simulator to have access to partial or complete witness of the potential language member[2]. Finally, in the quasi-adaptive setting, the language parameters may actually be generated by the CRS simulator and hence the simulator may have access to, say, the discrete logs of the language parameters, which can serve as further trapdoors.

Rather than considering these general settings, we focus on a simple partial-simulation setting, where (a) the CRS simulator can generate the language parameters itself and (b) the proof simulator when invoked with a word x is given an additional bit β, which we call the **membership bit**, that represents the information whether x is indeed a member or not.

The partial simulation world is required to be unbounded simulation-sound, and hopefully this should be easier to prove than usual unbounded simulation-soundness (given that its simulators have additional information). We also allow the partial simulation world to be sound with respect to a private verifier (this concept has been considered earlier in [JR12]), and this further leads to the possibility of easier and/or simpler constructions. A surprising property achievable under such a definition is that one can go back and forth between the partial-simulation world and the one-time full-simulation world even when simulating fake tuples.

Definition 2 (Dual-System Non-Interactive Proofs) *A Dual-system non-interactive proof system consists of PPT algorithms defined in three worlds as follows:*

Real World *consisting of:*

- *A pair of* **CRS generators** $(\mathsf{K}_0, \mathsf{K}_1)$*, where* K_0 *takes a unary string and produces an ensemble parameter* λ*. (The ensemble parameter* λ *is used to sample a witness-relation parameter* ρ *using* $\mathcal{D}_\lambda$ *in the security definition.) PPT algorithm* K_1 *uses* ρ *(and* λ*) to produce the real-world CRS* ψ.
- *A* **prover** P *that takes as input a CRS, a language member and its witness, a label, and produces a proof.*

[2]In case the proof simulator is being invoked on a non-language word, it is not immediately clear what this witness can be, unless we also define a language and a distribution for a super-language which includes the language under consideration as a subset.

- *A* **verifier** V *that takes as input a CRS, a word, a label, and a proof, and outputs a single bit.*

Partial-Simulation World *consisting of:*

- *A* **semi-functional CRS simulator** sfK_1 *that takes ensemble parameter λ as input and produces a witness relation parameter ρ, a semi-functional CRS σ, as well as two trapdoors τ and η. The first trapdoor is used by the proof simulator, and the second by the private verifier.*
- *A* **semi-functional simulator** sfSim *that takes a CRS, a trapdoor τ, a word, a membership-bit β, and a label, to produce a proof.*
- *A* **private verifier** pV *that takes a CRS, a trapdoor η, a word, a label, and a proof and outputs a single bit.*

One-time Full Simulation World *consisting of:*

- *A* **one-time full-simulation CRS generator** otfK_1*, that takes as input the ensemble parameter λ, the witness relation parameter ρ to produce a CRS and three trapdoors τ, τ_1 and η.*
- *A* **one-time full simulator** otfSim *that takes as input a CRS, a trapdoor τ_1, a word, a label, and produces a proof.*
- *A* **semi-functional verifier** sfV *that takes as input a CRS, a trapdoor η, a word, a label, a proof and outputs a bit. The adversaries also have access to the semi-functional simulator.*

Definition 3 (DSS-QA-NIZK) *The definition of the real-world components of a dual-system non-interactive proof to be complete and (computationally) sound are same as in QA-NIZK definition 1. Such a proof system is called a* **dual-system simulation-sound quasi-adaptive NIZK (DSS-QA-NIZK)** *for a collection of witness relations $\mathcal{R}_\lambda = \{R_\rho\}$, with parameters sampled from a distribution $\mathcal{D}$, if its real-world components are complete and (computationally) sound, and if for all non-uniform PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4)$ all of the following properties are satisfied (in all of the following probabilistic experiments, the experiment starts by setting λ as $\lambda \leftarrow \mathsf{K}_0(1^m)$):*

- **(Composable) Partial-ZK:**

$$\Pr[\rho \leftarrow \mathcal{D}_\lambda; \sigma \leftarrow \mathsf{K}_1(\lambda, \rho); \mathcal{A}_0(\sigma, \rho) = 1] \approx \Pr[(\rho, \sigma, \tau, \eta) \leftarrow \mathsf{sfK}_1(\lambda); \mathcal{A}_0(\sigma, \rho) = 1],$$

and

$$\Pr[(\rho, \sigma, \tau, \eta) \leftarrow \mathsf{sfK}_1(\lambda);\ \mathcal{A}_1^{\mathsf{P}(\sigma,\cdot,\cdot,\cdot),\ \mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot),\ \mathsf{V}(\sigma,\cdot,\cdot,\cdot)}(\sigma, \rho) = 1] \approx$$
$$\Pr[(\rho, \sigma, \tau, \eta) \leftarrow \mathsf{sfK}_1(\lambda);\ \mathcal{A}_1^{\mathsf{sfSim}^*(\sigma,\tau,\cdot,\cdot,\cdot),\ \mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot),\ \mathsf{pV}(\sigma,\eta,\cdot,\cdot,\cdot)}(\sigma, \rho) = 1],$$

where $\mathsf{sfSim}^*(\sigma, \tau, x, w, l)$ is defined to be $\mathsf{sfSim}(\sigma, \tau, x, \beta = 1, l)$ (i.e. witness is dropped, and membership-bit $\beta = 1$), **and** *the experiment aborts if either a call to the first oracle (i.e. P and sfSim^*) is with (x, w, l) s.t. $\neg R_\rho(x, w)$, or call to the second oracle is with an (x, β, l) s.t. $x \notin L_\rho$ or $\beta = 0$.*

- **Unbounded Partial-Simulation Soundness:**

$$\Pr[(\rho, \sigma, \tau, \eta) \leftarrow \mathsf{sfK}_1(\lambda);\ (x, l, \pi) \leftarrow \mathcal{A}_2^{\mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot),\ \mathsf{pV}(\sigma,\eta,\cdot,\cdot,\cdot)}(\sigma, \rho) :$$
$$((x \notin L_\rho) \vee \mathsf{V}(\sigma, x, l, \pi) = 0) \wedge \mathsf{pV}(\sigma, \eta, x, l, \pi) = 1] \approx 0.$$

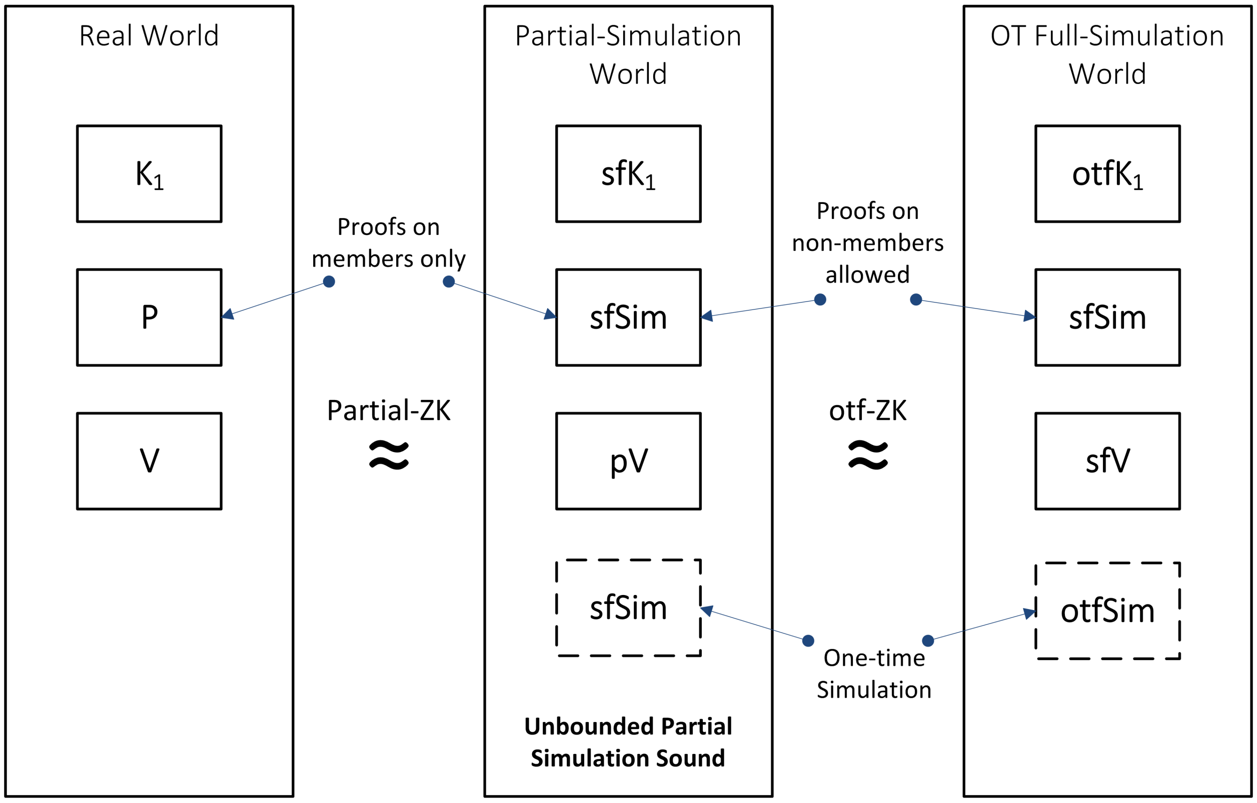

Figure 1: The three worlds of a DSS-QA-NIZK

- **One-time Full-ZK:**

$$\Pr[(\rho,\sigma,\tau,\eta) \leftarrow \mathsf{sfK}_1(\lambda); (x^*, l^*, \beta^*, s) \leftarrow \mathcal{A}_3^{\mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot),\, \mathsf{pV}(\sigma,\eta,\cdot,\cdot,\cdot)}(\sigma,\rho);$$
$$\pi^* \leftarrow \mathsf{sfSim}(\sigma,\tau,x^*,\beta^*,l^*) : \mathcal{A}_4^{\mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot),\, \mathsf{pV}(\sigma,\eta,\cdot,\cdot,\cdot)}(\pi^*, s) = 1] \approx$$
$$\Pr[\rho \leftarrow \mathcal{D}_\lambda;\ (\sigma,\tau,\tau_1,\eta) \leftarrow \mathsf{otfK}_1(\lambda,\rho); (x^*, l^*, \beta^*, s) \leftarrow \mathcal{A}_3^{\mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot),\, \mathsf{sfV}(\sigma,\eta,\cdot,\cdot,\cdot)}(\sigma,\rho);$$
$$\pi^* \leftarrow \mathsf{otfSim}(\sigma,\tau_1,x^*,l^*) : \mathcal{A}_4^{\mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot),\, \mathsf{sfV}(\sigma,\eta,\cdot,\cdot,\cdot)}(\pi^*, s) = 1],$$

where the experiment aborts if either in the call to the first oracle, or in the (x^*, β^*) *produced by* $\mathcal{A}_3$*, the membership-bit provided is not correct for* L_ρ*,* **or** *if* $\langle x^*, l^*, \pi^* \rangle$ *is queried to* sfV/pV*. Here* s *is a state variable.*

The three worlds and the properties of a DSS-QA-NIZK are depicted in Figure 1.

Remark 1. In the partial-simulation soundness definition, there is *no restriction* of x, l, π being *not* the same as that obtained from a call to the first oracle sfSim.

Remark 2. Note that in the partial-ZK definition, the calls to the prover are restricted to ones satisfying the relation. However, the calls to the simulator sfSim in the one-time full-ZK definition are only restricted to having the correct membership bit β.

Remark 3. It can be shown that sfSim generated proofs on words (whether members or not) are accepted by real-world verifier V (with semi-functional CRS). Of course, the private verifier pV will *even* reject proofs generated by sfSim on non-language words. This justifies the name "semi-functional simulator". See Lemma 11 in Appendix A for a precise claim and proof.

It can also be shown that the semi-functional verifier sfV is still complete, i.e. it accepts language members and proofs generated on them by $\mathsf{P}(\sigma, \cdot, \cdot, \cdot)$ (with σ generated by otfK_1). As opposed to P and pV, it may no longer be sound. This justifies the name "semi-functional verifier" a la Waters' dual-system IBE construction. However, if the one-time full-ZK property holds statistically, it can be shown that the semi-functional verifier is sound in the one-time full-simulation world. See Lemma 12 in Appendix A for a precise statement.

Remark 4. The composable partial-ZK and unbounded partial-simulation soundness imply that that the system is true-simulation-sound [Har11] w.r.t. the semi-functional simulator. More precisely, the following lemma holds.

Lemma 4 (true-simulation-soundness) *For a DSS-QA-NIZK, for all PPT $\mathcal{A}$,*

$$\Pr[(\rho, \sigma, \tau, \eta) \leftarrow \mathsf{sfK}_1(\lambda);\ (x, l, \pi) \leftarrow \mathcal{A}^{\mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot)}\ (\sigma, \rho)\ :\ (x \notin L_\rho)\ \wedge\ \mathsf{V}(\sigma, x, l, \pi) = 1]\ \approx 0,$$

where the experiment aborts if $\mathcal{A}$ invokes the oracle with some (y, β, l) , s.t. $y \notin L_\rho$ or $\beta = 0$.

The lemma is proved in Appendix A.

4 DSS-QA-NIZK for Linear Subspaces

In this section we show that languages that are linear subspaces of vector spaces of hard bilinear groups have very short dual-system simulation sound QA-NIZK. In fact, under the symmetric-external Diffie-Hellman (SXDH) assumption, such proofs only require two group elements, regardless of the subspace. It was shown in [JR14] that such subspaces have a QA-NIZK proof of just one group element (under SXDH assumption). Our construction essentially shows that with one additional group element, one can make the QA-NIZK dual-system simulation-sound. We will actually show a more general construction which is more widely applicable, and does not even refer to bilinear groups or linear subspaces. Informally speaking, the requirement for such a general construction for parameterized languages is that each language have a 2-universal projective hash proof system and the augmented language with this hash proof attached have a QA-NIZK proof system with statistical zero-knowledge. A few other properties of the QA-NIZK are required for this construction, and we show that such properties already hold for the construction of [JR14]. Since for linear subspaces, 2-universal projective hash proofs are rather easy to obtain, the general construction along with the QA-NIZK of [JR14] allows us to obtain a short DSS-QA-NIZK for linear subspaces. Apart from abstracting the main ideas involved in the DSS-QA-NIZK construction for linear subspaces, the general construction's wider applicability also allows us to extend our results to linear subspaces with tags.

We start this section by briefly reviewing projective hash proofs [CS02], and their extensions to distributions of languages, as they are extensively used in the rest of the section.

Projective Hash Proof System. For a language L, let X be a superset of L and let $H = (H_k)_{k \in K}$ be a collection of (hash) functions indexed by K with domain X and range another set Π. The hash function family is generalized to a notion of *projective hash function family* if there is a set S of projection keys, and a projection map $\alpha: K \rightarrow S$, and further the action of H_k on subset L of X is completely determined by the projection key $\alpha(k)$. Finally, the projective hash function family is defined to be ϵ-**universal$_2$** is for all $s \in S$, $x, x^* \in X$, and $\pi, \pi^* \in \Pi$ with $x \notin L \cup \{x^*\}$, the following holds:

$$\Pr[H_k(x) = \pi \mid H_k(x^*) = \pi^* \wedge \alpha(k) = s] \leq \epsilon.$$

A projective hash function family is called ϵ-**smooth** if for all $x \in X \setminus L$, the statistical distribution between the following two distributions is ϵ: sample k uniformly from K and π' uniformly from Π; the first distribution is given by the pair $(\alpha(k), H_k(x))$ and the second by the pair $(\alpha(k), \pi')$. For languages defined by a witness-relation R, the projective hash proof family constitutes a *projective hash proof system* (PHPS) if α, H_k, and another *public evaluation function* $\hat{H}$ that computes H_k on $x \in L$, given a witness of x and *only* the projection key $\alpha(k)$, are all efficiently computable. An efficient algorithm for sampling the key $k \in K$ is also assumed.

The above notions can also incorporate labels. In an *extended PHPS*, the hash functions take an additional input called *label*. The public evaluation algorithm also takes this additional input called label. All the above notions are now required to hold for each possible value of label. The extended PHPS is now defined to be ϵ-**universal$_2$** is for all $s \in S$, $x, x^* \in X$, all labels l and l^*, and $\pi, \pi^* \in \Pi$ with $x \notin L$ and $(x, l) \neq (x^*, l^*)$, the following holds: $\Pr[H_k(x, l) = \pi \mid H_k(x^*, l^*) = \pi^* \wedge \alpha(k) = s] \leq \epsilon$.

Since, we are interested in distributions of languages, we extend the above definition to distribution of languages. So consider a parametrized class of languages $\{L_\rho\}_{\rho \in \mathsf{Lpar}}$ with the parameters coming from an associated parameter language Lpar. Assume that all the languages in this collection are subsets of X. Let H as above be a collection of hash functions from X to Π. We say that the hash family is a projective hash family if for all L_ρ, the action of H_k on L_ρ is determined by $\alpha(k)$. Similarly, the hash family is ϵ-universal$_2$ (ϵ-smooth) for $\{L_\rho\}_{\rho \in \mathsf{Lpar}}$ if for all languages L_ρ the ϵ-universal$_2$ (resp. ϵ-smooth) property holds.

Intuition for the Construction. The main idea of the construction is to first attach (as a proof component) a universal$_2$ and smooth projective hash proof T. The DSS-QA-NIZK is then just (T, π) , where π is a QA-NIZK proof of the original language augmented with hash proof T. So, why should this work? First note that the smooth projective hash function is a designated-verifier NIZK, and hence this component T is used in private verification. Secondly, since it is universal$_2$, its soundness will hold even when the Adversary gets to see the projection key $\alpha(k)$ plus one possibly fake hash proof (i.e. $H_k(x)$, where x not in the language).

We will assume in our general construction that the parameterized language is such that the simulator can sample the language parameters along with auxiliary information that allows it to easily verify a language member. For example, this auxiliary information can be discrete logs of the language parameters. The idea of obtaining partial-ZK and unbounded partial-simulation soundness is then pretty simple. The proof simulation of T is easy to accomplish given the hash keys and, crucially, the correct membership-bit. In fact, if the membership-bit is false, T can just be set randomly (by smoothness). The simulation of π part of the proof is done using the QA-NIZK simulation trapdoor. The private verification is done as conjunction of three separate checks: (a) using the auxiliary information, (b) using the hash proof and (c) using the real-world verifier.

Now, in the one-time full simulation, the auxiliary information is not available, but the semi-functional verifier can still use hash keys. Further, we can have one bad use of keys (in full simulation of one proof. Since the oracle calls to semi-functional simulator sfSim are restricted to having correct membership-bit, they do not yield any additional information about the hash keys.

Requirements of the General Construction. Consider a parameterized class of languages $\{L_\rho\}_{\rho \in \mathsf{Lpar}}$, and a probability distribution $\mathcal{D}$ on Lpar. Assume that this class has a projective hash proof system as above. Let R_ρ be the corresponding witness relation of L_ρ. Now consider the augmented witness-relation $R^*_{\rho,s}$ defined as follows (for $\rho \in \mathsf{Lpar}$ and $s \in S$):

$$R^*_{\rho,s}(\langle x, T, l\rangle, w) \equiv (R_\rho(x, w) \ \wedge \ T \overset{?}{=} \hat{H}(s, \langle x, l\rangle, w)).$$

Note, the witness remains the same for the augmented relation. Since H is a projective hash function, it follows that for $s = \alpha(k)$, the corresponding augmented language is $L^*_{\rho,s} = \{(x, T, l) \mid x \in L_\rho \wedge T \overset{?}{=} H_k(x, l)\}$. Let the distribution $\mathcal{D}'$ on pairs (ρ, s) be defined by sampling ρ according to $\mathcal{D}$ and sampling k uniformly from K, and setting $s = \alpha(k)$. We remark that the language parameters of the augmented language include projection keys s (instead of keys k) because it is crucial that the CRS simulator in the quasi-adaptive NIZK gets only the projection key s (and not k).

We will also assume that the distribution $\mathcal{D}$ on Lpar is efficiently *witness sampleable* which is defined by requiring that there are two efficient (probabilistic) algorithms E_1, E_2 such that E_1 can sample ρ from $\mathcal{D}$ along with auxiliary information ψ (which can be thought of as witness of ρ in the language Lpar), and E_2 can decide w.h.p. if a potential language member x is in L_ρ given ρ and ψ, where the probability is defined over choice of ρ according to $\mathcal{D}$ and the internal coins of E_2.

Finally, we need a few additional properties of QA-NIZK proofs (section 2) that we now define. We will later show that the single group element QA-NIZK construction for linear-subspaces of [JR14] already satisfies these properties.

Definition 5 *There are various specializations of QA-NIZK of interest:*

- *The QA-NIZK (Section 2) is said to have* **composable zero-knowledge** *[GS08] if the CRS are indistinguishable in the real and simulation worlds, and the simulation is indistinguishable even if the adversary is given the trapdoor. More precisely, for all PPT adversary $\mathcal{A}_1, \mathcal{A}_2$,*
  $\Pr[\textsc{crs} \leftarrow \mathsf{crsgen}(\lambda, \rho) : \mathcal{A}_1(\textsc{crs}, \rho) = 1] \ \approx \ \Pr[(\textsc{crs}, \mathsf{trap}) \leftarrow \mathsf{crs\text{-}sim}(\lambda, \rho) : \mathcal{A}_1(\textsc{crs}, \rho) = 1]$,
  and
  $\Pr[(\textsc{crs}, \mathsf{trap}) \leftarrow \mathsf{crs\text{-}sim}(\lambda, \rho) : \mathcal{A}_2^{\mathsf{prover}(\textsc{crs},\cdot,\cdot,\cdot)}(\textsc{crs}, \rho, \mathsf{trap}) = 1] \approx$
  $\Pr[(\textsc{crs}, \mathsf{trap}) \leftarrow \mathsf{crs\text{-}sim}(\lambda, \rho) : \mathcal{A}_2^{\mathsf{sim}^*(\textsc{crs},\mathsf{trap},\cdot,\cdot,\cdot)}(\textsc{crs}, \rho, \mathsf{trap}) = 1]$,
  where $\mathcal{A}_2$ is restricted to calling the oracle only on (x, w, l) with $(x, w) \in R_\rho$.

- *The QA-NIZK is called* **true-simulation-sound** *[Har11] if the verifier is sound even when an adaptive adversary has access to simulated proofs on language members. More precisely, for all PPT $\mathcal{A}$,* $\Pr[(\textsc{crs}, \mathsf{trap}) \leftarrow \mathsf{crs\text{-}sim}(\lambda, \rho);\ (x, l, \pi) \leftarrow \mathcal{A}(\textsc{crs}, \rho)^{\mathsf{sim}(\textsc{crs},\mathsf{trap},\cdot,\cdot)} : x \notin L_\rho \ \wedge \ \mathsf{ver}(\textsc{crs}, x, l, \pi) = 1] \approx 0$, *where the experiment aborts if the oracle is called with some $y \notin L_\rho$.*

- *The simulator is said to generate* **unique acceptable proofs** *if for all x, all labels l, and all proofs π^*,* $\Pr[(\textsc{crs}, \mathsf{trap}) \leftarrow \mathsf{crs\text{-}sim}(\lambda, \rho);\ \pi \leftarrow \mathsf{sim}(\textsc{crs}, \mathsf{trap}, x, l) : (\pi^* \neq \pi)$ *and* $\mathsf{ver}(\textsc{crs}, x, l, \pi^*) = 1] \approx 0$.

General Construction. We now show that given:

1. An ϵ-smooth and ϵ-universal$_2$ (labeled) projective hash proof system for the collection $\{L_\rho\}_{\rho \in \mathsf{Lpar}}$, and
2. A composable zero-knowledge, true-simulation-sound QA-NIZK Q= (pargen, crsgen, prover, ver, $\mathsf{crs\text{-}sim}$, sim) for the augmented parameterized language $L^*_{\rho,s}$ with probability distribution $\mathcal{D}'$, such that the simulator *generates unique acceptable proofs*, and
3. Efficient algorithms (E_1, E_2) s.t. $\mathcal{D}$ is efficiently witness-sampleable using (E_1, E_2), and
4. An efficient algorithm E_3 to sample uniformly from Π,

one can construct a DSS-QA-NIZK for $\{L_\rho\}_{\rho \in \mathsf{Lpar}}$ with probability distribution $\mathcal{D}$. We first give the construction, and then prove the required properties. The QA-NIZK Q *need not take any labels as input.* The various components of the dual-system non-interactive proof system Σ are as follows.

Real World consisting of:

- The algorithm K_0 takes a unary string 1^m as input and generates parameters λ using pargen of Q on 1^m. The CRS generation algorithm K_1 uses crsgen of Q and produces the CRS as follows: it takes λ and the language parameter ρ, and first samples k uniformly from K_λ (recalling that the hash function families are ensembles, one for each λ). It then outputs the CRS to be the pair $(\mathsf{crsgen}(\lambda, \langle \rho, \alpha(k) \rangle), \alpha(k))$.
- The **prover** P takes a CRS (σ, s), input x, witness w, and label l and outputs the proof to be $(T,\, W)$ where T is computed using the public evaluation algorithm $\hat{H}$ as $\hat{H}(s, \langle x, l \rangle, w)$ and $W = \mathsf{prover}(\sigma, \langle x, T, l \rangle, w)$.
- The **verifier** V on input CRS $= (\sigma', s)$, x, l, and proof (T, W), returns the value $\mathsf{ver}(\sigma', \langle x, T, l \rangle, W)$ (using ver of Q).

Partial-Simulation World consisting of:

- The **semi-functional CRS simulator** sfK_1 takes λ as input and samples (ρ, ψ) using E_1, and also samples k uniformly from K_λ. It then uses $\mathsf{crs\text{-}sim}$ of Q, and key projection algorithm α to generate the CRS σ as follows: Let $(\sigma', \mathsf{trap}) = \mathsf{crs\text{-}sim}(\lambda, \langle \rho, \alpha(k) \rangle)$. The CRS σ is then the pair $(\sigma', \alpha(k))$. sfK_1 also outputs k, trap as proof simulator trapdoors τ, and ρ, ψ, k as private verifier trapdoors η.
- The **semi-functional simulator** sfSim uses trapdoors k, trap to produce a (partially-simulated) proof for a potential language member x, a label l and a binary bit β using sim of Q as follows: if $\beta = 1$, output
$$T = H_k(x, l),\ W = \mathsf{sim}(\sigma, \mathsf{trap}, \langle x, T, l \rangle),$$
else sample π' at random from Π (using E_3) and output
$$T = \pi'\ ,\ W = \mathsf{sim}(\sigma, \mathsf{trap}, \langle x, T, l \rangle).$$
This proof is partially simulated as it uses the bit β.

- The **private verifier** pV uses trapdoors (ρ, ψ, k) to check a potential language member x, label l and a proof T, W as follows: it outputs 1 iff (a) E_2 using ρ and ψ confirms that x is in L_ρ, and (b) $H_k(x, l)$ computes to be equal to T, and (c) verifier of Q accepts, i.e. $\mathsf{ver}(\sigma, \langle x, T, l\rangle, W) = 1$.

One-time Full Simulation World consisting of:

- The **one-time full-simulation CRS generator** otfK_1 takes as input λ and language parameter ρ, and using $\mathsf{crs\text{-}sim}$ of Q outputs σ as follows: first it samples k uniformly from K_λ. Let $(\sigma', \mathsf{trap}) = \mathsf{crs\text{-}sim}(\lambda, \langle \rho, \alpha(k)\rangle)$. Then $\sigma = (\sigma', \alpha(k))$. otfK_1 also outputs k, trap as proof simulator trapdoors τ and τ_1, and outputs k as private verifier trapdoor η.
- The **one-time full simulator** otfSim takes as input the trapdoors k, trap and a potential language member x and a label l to produce a proof as follows:

$$T = H_k(x, l), \quad W = \mathsf{sim}(\sigma, \mathsf{trap}, \langle x, T, l\rangle).$$

- The **semi-functional verifier** sfV uses trapdoors k to verify a potential language member x, a label l and a proof T, W as follows: output 1 iff (a) $H_k(x, l)$ computes to be same as T, and (b) $\mathsf{ver}(\sigma, \langle x, T, l\rangle, W) = 1$.

Theorem 6 *For a parameterized class of languages $\{L_\rho\}_{\rho \in \mathsf{Lpar}}$ with probability distribution $\mathcal{D}$, if the above four conditions hold for projective hash family H, QA-NIZK Q, and efficient algorithms E_1, E_2, E_3, then the above dual-system non-interactive proof system Σ is a DSS-QA-NIZK for $\{L_\rho\}_{\rho \in \mathsf{Lpar}}$ with probability distribution $\mathcal{D}$.*

Remark. in Appendix B.1 we instantiate the general construction for linear subspaces of vector spaces of hard bilinear groups. As a corollary, it follows that under the SXDH assumption the Diffie-Hellman (DH) language has a DSS-QA-NIZK with only two group elements.

Due to space limitations, we will focus on only the proof of one-time zero-knowledge (otzk) property, as that is the most non-trivial proof. Indeed, this property is a significant generalization of the usual dual-system technique employed in IBE constructions because although in otzk only one proof needs to be fully simulated (i.e. without its membership bit being available), all the private verifier calls in the partial-simulation world need to be simulated in the otzk world without the quasi-adaptive trapdoors (i.e. trapdoor obtained by witness-sampling the language parameters). Recall, in the IBE construction the ciphertext is the counterpart of our verifier, and the IBE private keys are the QA-NIZK proofs. Thus, in IBE only a single ciphertext needs to be simulated when the different private keys are being "fixed" one-by-one by otzk simulation.

The detailed proof of all other properties is given in Appendix B. The main idea of the proof of these properties is already sketched earlier in this section.

Lemma 7 *In the context of Theorem 6, let the maximum probability that the simulator of Q does not generate unique acceptable proofs be δ. Let H be an ϵ-smooth and ϵ-universal$_2$ (labeled) projective hash proof system for the collection $\{L_\rho\}_{\rho \in \mathsf{Lpar}}$. Let M be the number of calls to the second oracle (verifier) by $\mathcal{A}_3$ and $\mathcal{A}_4$ combined in the two experiments of the one-time full-ZK property of DSS-QA-NIZK Σ. Then the maximum statistical distance (over all Adversaries) between the views of the adversaries $(\mathcal{A}_3, \mathcal{A}_4)$ in these two experiments, denoted $\textsc{dist}^{otzk}(\Sigma)$, is at most $(\epsilon + \delta) * (1 + M)$.*

Proof: We will show that the one-time full-ZK property holds statistically. We will define a sequence of experiments and show that the view of the adversary is statistically indistinguishable in every two consecutive experiments. The first experiment $\mathbf{H}_0$ is identical to the partial-simulation world. First, note that ρ is identically generated using $\mathcal{D}$ in both worlds. Next, note that the CRS σ and trapdoors τ generated by sfK_1 is identically distributed to the CRS σ and both the trapdoors τ and τ_1 generated by otfK_1.

The next experiment $\mathbf{H}_1$ is identical to $\mathbf{H}_0$ except that on $\mathcal{A}_3$ supplied input (x^*, l^*, β^*) the proof π^* generated by sfSim is replaced by proof generated by otfSim. If β^* provided by $\mathcal{A}_3$ is not the valid membership bit for x^* then both experiments abort. So, assume that β^* is the correct membership bit. In case $\beta^* = 1$, both sfSim and otfSim behave identically. When $\beta^* = 0$, the random T^* produced by sfSim is identically distributed to the T^* generated by $H_k(x^*, l^*)$ since H is assumed to be smooth.

The next experiment $\mathbf{H}_2$ is identical to $\mathbf{H}_1$ except that the second oracle is replaced by sfV (from being pV). In order to show that the view of the adversary is indistinguishable in experiments $\mathbf{H}_2$ and $\mathbf{H}_1$, we define several hybrid experiments $\mathbf{H}_{1,i}$ (for $0 \leq i \leq N$, where N is the total number of calls to the second-oracle by $\mathcal{A}_3$ and $\mathcal{A}_4$ combined). Experiment $\mathbf{H}_{1,0}$ is identical to $\mathbf{H}_1$, and the intermediate experiments are defined inductively, by modifying the response of one additional second-oracle call starting with the last (N-th) second-oracle call, and ending with the changed response of the first second-oracle call. The last hybrid experiment $\mathbf{H}_{1,N}$ will then be same as $\mathbf{H}_2$. The second-oracle call response in experiment $\mathbf{H}_{1,i+1}$ differs only in the $(N-i)$-th second-oracle call response in $\mathbf{H}_{1,i}$. In the latter experiment, this call is still served as in $\mathbf{H}_1$ (i.e. using pV). In the former experiment $\mathbf{H}_{1,i+1}$, the $(N-i)$-th call is responded to as defined in $\mathbf{H}_2$ above (i.e. using sfV).

To show that the view of the adversary is statistically indistinguishable in $\mathbf{H}_{1,i}$ and $\mathbf{H}_{1,i+1}$, first note that the view of the adversary ($\mathcal{A}_3$ and A_4 combined) till it's $(N-i)$-th call in both experiments is identical. Moreover, as we next show, the dependence on k of this partial view (i.e. till the $(N-i)$-th call) is limited to $\alpha(k)$ and at most one evaluation of H_k (by otfSim) on an input that is not in L_ρ. To start with, the CRS generated by sfK_1 depends only on $\alpha(k)$. Next, the first oracle sfSim produces T using H_k on its input only if the membership bitβ is 1 and correct, and since H is projective this hash value is then completely determined by $\alpha(k)$. Finally, all calls to the second oracle till the $(N-i)$-th call are still served using pV, and again using the projective property of H, it is clear that the conjunct (b) in pV can be computed using only $\alpha(k)$, because for non L_ρ members, the conjunct (a) is already false, and hence (b) is redundant.

Now, the difference in the $(N-i)$-th call is that the conjunct (a) of pV is missing in sfV. Let x, l, T, W be the input supplied by the Adversary to this call. If $H_k(x, l)$ is not equal to the supplied T, then both pV and sfV return 0. So, suppose $H_k(x, l)$ is equal to T, and yet x is not in L_ρ, i.e. conjunct (a) of pV is false. First, if this input x, l, T, W is same as (x^*, l^*, T^*, W^*) associated with the one-time call to otfSim, then the experiment aborts. Thus, we can assume that this is a different input. If (x, l) is same as (x^*, l^*), then $(T, W) \neq (T^*, W^*)$. Now, by construction (i.e. by definition of otfSim) $T^* = H_k(x^*, l^*)$, and hence either $T \neq H_k(x, l)$ which is not possible by hypothesis, or $(x, l, T) = (x^*, l^*, T^*)$ and $W \neq W^*$. But, W^* is proof generated by the simulator of Q, and since the simulator of Q generates unique acceptable proofs (by assumption), the verifier ver of Q rejects (x, l, T, W), and thus both pV and sfV return 0.

On the other hand, if $(x, l) \neq (x^*, l^*)$ then by the ϵ-universal$_2$ property of H, the probability of T being same as $H_k(x, l)$ is at most ϵ. Thus, both pV and sfV return 0. That completes the induction step, and thus the view of the adversary in experiments $\mathbf{H}_1$ and $\mathbf{H}_2$ is statistically indistinguishable.

The next experiment $\mathbf{H}_3$ is identical to $\mathbf{H}_2$ except that the CRS is generated using otfK_1. The only difference is that the (verifier) trapdoor does not include ρ, ψ. But, since the second oracle is served by sfV and it does not need ρ, ψ, the experiment $\mathbf{H}_3$ is well-defined and statistically indistinguishable from $\mathbf{H}_2$, Further, $\mathbf{H}_3$ is identical to the one-time simulation world, and that completes the proof.

The statistical distance between the views of the adversaries $(\mathcal{A}_3, \mathcal{A}_4)$ in $\mathbf{H}_0$ and $\mathbf{H}_3$ is at most $(\epsilon + \delta) * (1 + M)$. □ □

5 Keyed-Homomorphic CCA Encryption

Keyed-Homomorphic Encryption is a primitive, first developed in [EHO+13], which allows homomorphic operations with a restricted evaluation key, while preserving different flavors of semantic security depending on whether access to the evaluation key is provided or not. For an adversary not having access to the evaluation key, the homomorphic operation should not be available and this is ensured by requiring CCA security. However, if an adversary comes into possession of the evaluation key, CCA security can no longer be preserved and thus weaker forms of security, such as CCA1, are required. In [LPJY14], the authors gave improved constructions for multiplicative homomorphism with better security guarantees.

A **KH-PKE** scheme consists of algorithms $(KeyGen, Enc, Dec, Eval)$, where the first three are familiar from public-key encryption, and KeyGen generates a public key pk, a decryption key sk_d and an Eval key sk_h. Algorithm Eval takes two ciphertexts and returns a ciphertext or $\perp$. Detailed definitions can be found in Appendix C. The scheme is said to be correct if (i) for Enc we have $Dec(sk_d, Enc(pk, M)) = M$, where sk_d is the secret decryption key, and (ii) for Eval we have $Dec(sk_d, Eval(sk_h, C_1, C_2)) = Dec(sk_d, C_1) \odot Dec(sk_d, C_2)$, where $\odot$ is a binary operation on plaintexts, and if any operand of $\odot$ is $\perp$ then the result is $\perp$. The KH-PKE scheme is defined to be **KH-CCA** secure by a usual public-key CCA experiment with the following twists: the challenger maintains a set D of ciphertexts dependent on the challenge ciphertext (via Eval); decryption queries are not allowed on ciphertexts in D. Further, an adversary $\mathcal{A}$ can adaptively ask for sk_h, which we call the *reveal event*. After the reveal event, the Eval oracle is not available. Similarly, decryption is not available after $\mathcal{A}$ has both requested sk_h and obtained the challenge ciphertext, in any order. Again, detailed definitions can be found in Appendix C.

Construction. We present a construction of a KH-CCA secure KH-PKE encryption scheme with multiplicative homomorphism which utilizes our general DSS-QA-NIZK construction for the Diffie-Hellman (DH) language. In fact, if we assume that the adversary never invokes RevHK, we can prove security generically assuming any DSS-QA-NIZK (with statistical one-time full-ZK) for the DH language. When the adversary invokes RevHK, the partial-simulation trapdoor is revealed to the Adversary, and hence the one-time full-ZK property of DSS-QA-NIZK may not hold. Thus, we a need a stronger notion of DSS-QA-NIZK that incorporates the reveal event, and includes an additional requirement that the semi-functional verifier remains sound as before. Using this stronger notion, we can prove generic security of the KH-PKE scheme even with RevHK, and we further show that our general construction of Section 4 continues to satisfy this stronger property.

We start with the observation that a standard El Gamal encryption scheme $(\mathbf{g}^x, m \cdot \mathbf{f}^x)$ is multiplicatively homomorphic, but is not CCA secure due to the exact same reason. The main idea of our construction is as follows. The ciphertexts include an El-Gamal encryption of the message M, say $\mathbf{g}^r, M \cdot \mathbf{g}^{kr}$ for a public key $\mathbf{g}^k$. The public key also consists of a member $\mathbf{g}^a$, and the ciphertext also include $\mathbf{g}^{ar}$ (we refer to this triple in the ciphertext as *augmented El-Gamal encryption*). It is well-known [JR12] that if a one-time simulation-sound NIZK proof of $\mathbf{g}^r$ and $\mathbf{g}^{ar}$ being of the correct form is also included in the ciphertext then it becomes a publicly-verifiable CCA2-secure encryption scheme. In our keyed-homomorphic construction, we include a DSS-QA-NIZK for $\mathbf{g}^r$ and $\mathbf{g}^{ar}$ being of the correct form (i.e. being a DH tuple). Although the DSS-QA-NIZK itself is not homomorphic, we can take advantage of the corresponding Semi-Functional Simulator sfSim and simulate the proof of a multiplicatively generated (augmented) El Gamal encryption when computing a homomorphic evaluation.

So, given a dual-system non-interactive proof Σ, consider the following algorithms for a KH-PKE scheme $\mathcal{P}$:

KeyGen: Generate $\mathbf{g}, a, k$ randomly. Use sfK_1 of Σ to get CRS σ and trapdoors τ and η, and language parameters $\rho = (\mathbf{g}, \mathbf{g}^a)$. Set $pk = (\mathbf{g}, \mathbf{g}^a, \mathbf{g}^k, \sigma)$, $sk_h = \tau$, $sk_d = k$.

Enc: Given plaintext m, generate $w \leftarrow \mathbb{Z}_q$ and compute (using P of Σ) $c := (\mathbf{g}^w, \mathbf{g}^{aw}, \gamma, \mathsf{P}(\sigma, (\mathbf{g}^w, \mathbf{g}^{aw}), w, l = \gamma))$, where $\gamma := m \cdot \mathbf{g}^{kw}$.

Dec: Given ciphertext $c = (\rho, \hat{\rho}, \gamma, \pi)$, first check if $\mathsf{V}(\sigma, \pi, (\rho, \hat{\rho}), \gamma)$ of Σ holds, then compute $m := \gamma/\rho^k$.

Eval (multiplicative): Given ciphertexts $c_1 = (\rho_1, \hat{\rho}_1, \gamma_1, \pi_1)$ and $c_2 = (\rho_2, \hat{\rho}_2, \gamma_2, \pi_2)$, first check if $\mathsf{V}(\sigma, \pi_i, (\rho_i, \hat{\rho}_i), \gamma_i)$ of Σ holds for all $i \in \{1, 2\}$. Then compute: $\rho = \rho_1\rho_2\rho_3$, $\hat{\rho} = \hat{\rho}_1\hat{\rho}_2\hat{\rho}_3$, $\gamma = \gamma_1\gamma_2\gamma_3$, where $\langle \rho_3, \hat{\rho}_3, \gamma_3 \rangle$ is a fresh random tuple obtained by picking r at random and setting the tuple to be $\langle \mathbf{g}^r, (\mathbf{g}^a)^r, (\mathbf{g}^k)^r \rangle$. Then compute $\pi := \mathsf{sfSim}(\sigma, \tau, (\rho, \hat{\rho}), \beta = 1, l = \gamma)$ using sfSim of Σ. Output ciphertext $c := (\rho, \hat{\rho}, \gamma, \pi)$.

Theorem 8 (Security of Construction) *The above algorithms P= (KeyGen, Enc, Dec, Eval) constitute a KH-CCA secure Keyed-Homomorphic Public Key Encryption scheme with multiplicative homomorphism, if Σ is a DSS-QA-NIZK for the parameterized Diffie-Hellman language (with language parameters distributed randomly) and RevHK is not available.*

The main idea of the proof of the above theorem is similar to proofs of CCA2-secure public key encryption schemes using alternate decryption. In other words, the ciphertext can be decrypted as $m := \gamma/\rho^k$, or as $m := \gamma/(\rho^{k_0}\hat{\rho}^{k_1})$, where $k = k_0 + ak_1$. But, this requires that the ciphertext has correct $\hat{\rho}$ component, i.e. $\hat{\rho} = \rho^a$. The ciphertexts include a NIZK for this purpose, but the NIZK needs to be simulation-sound. Additional complication arises because of dependent ciphertexts. To handle this, we first build an intermediate experiment where all dependent ciphertexts are generated using fresh random El Gamal tuples. Indistinguishability of such an intermediate experiment from the KH-CCA experiment is shown inductively, by carefully employing one-time full-ZK and partial-simulation unbounded simulation soundness. The theorem is proved in detail in Appendix C.1. The Adversary's advantage in the KH-CCA security game is at most $(8L+1) \cdot \mathrm{ADV}_{\mathrm{DDH}} + O(L/q)$, where L is the total number of calls to Eval.

The more general theorem (with RevHK) is stated and proved in Appendix C.2. Under the SXDH assumption, the above construction leads to ciphertexts of size only five group elements. Further, using an *augmented Diffie Hellman language* (augmented with a smooth hash proof of

DH tuple) and its DSS-QA-NIZK, we also extend our result to get CCA1-security despite the key being revealed (see Appendix C.2). The resulting scheme has KH-PKE ciphertexts of size six group elements.

6 Single-Round UC Password-Based Key Exchange

The essential elements of the Universal Composability framework can be found in [Can01]. In the following, we adopt the definition for password-based key exchange (UC-PAKE) from Canetti et al [CHK+05].

6.1 UC-PAKE Definition

Functionality $\mathcal{F}_{\mathbf{pake}}$

The functionality $\mathcal{F}_{\text{PAKE}}$ is parameterized by a security parameter k. It interacts with an adversary S and a set of parties via the following queries:

Upon receiving a query $(\mathsf{NewSession}, sid, P_i, P_j, pw, role)$ **from party** P_i**:**
Send $(\mathsf{NewSession}, sid, P_i, P_j, role)$ to S. In addition, if this is the first NewSession query, or if this is the second NewSession query and there is a record (P_j, P_i, pw'), then record (P_i, P_j, pw) and mark this record fresh.

Upon receiving a query $(\mathsf{TestPwd}, sid, P_i, pw')$ **from the adversary** S**:**
If there is a record of the form (P_i, P_j, pw) which is fresh, then do: If $pw = pw'$, mark the record compromised and reply to S with "correct guess". If $pw \neq pw'$, mark the record interrupted and reply with "wrong guess".

Upon receiving a query $(\mathsf{NewKey}, sid, P_i, sk)$ **from** S**, where** $|sk| = k$**:**
If there is a record of the form (P_i, P_j, pw), and this is the first NewKey query for P_i, then:

- If this record is compromised, or either P_i or P_j is corrupted, then output (sid, sk) to player P_i.
- If this record is fresh, and there is a record (P_j, P_i, pw') with $pw' = pw$, and a key sk' was sent to P_j, and (P_j, P_i, pw) was fresh at the time, then output (sid, sk') to P_i.
- In any other case, pick a new random key sk' of length k and send (sid, sk') to P_i.

Either way, mark the record (P_i, P_j, pw) as completed.

Upon receiving $(\mathsf{Corrupt}, sid, P_i)$ **from** S**:** if there is a (P_i, P_j, pw) recorded, return pw to S, and mark P_i corrupted.

Figure 2: The password-based key-exchange functionality $\mathcal{F}_{\text{PAKE}}$

Just as in the normal key-exchange functionality, if both participating parties are not corrupted, then they receive the same uniformly distributed session key and the adversary learns nothing of the key except that it was generated. However, if one of the parties is corrupted, then the adversary determines the session key. This power to the adversary is *also* given in case it succeeds in guessing the parties' shared password. Participants also detect when the adversary makes an unsuccessful attempt. If the adversary makes a wrong password guess in a given session, then the session is marked interrupted and the parties are provided random and independent session keys. If however the adversary makes a successful guess, then the session is marked compromised, and the adversary is allowed to set the session key. If a session remains marked fresh, meaning that it is neither interrupted nor compromised. uncorrupted parties conclude with both parties receiving the same, uniformly distributed session key.

The formal description of the UC-PAKE functionality $\mathcal{F}_{\text{PAKE}}$ is given in Figure 2.

If a party is corrupted, the ideal world adversary is provided with the password. Moreover, if a party is corrupted after its peer has already issued a session key, and the party has not yet issued a session key, then the Adversary is given the session key.

The real-world protocol we provide is also shown to be secure when different sessions use the same common reference string (CRS) To achieve this goal, we consider the *universal Composability with joint state* (JUC) formalism of Canetti and Rabin [CR03]. This formalism provides a "wrapper layer" that deals with "joint state" among different copies of the protocol. In particular, defining a functionality $\mathcal{F}$ also implicitly defines the multi-session extension of $\mathcal{F}$(denoted by $\hat{\mathcal{F}}$): $\hat{\mathcal{F}}$ runs multiple independent copies of $\mathcal{F}$, where the copies are distinguished via sub-session IDs ssid. The JUC theorem [CR03] asserts that for any protocol π that uses multiple independent copies of $\mathcal{F}$, composing π instead with a single copy of a protocol that realizes $\hat{\mathcal{F}}$, preserves the security of π.

6.2 UC-RPAKE Definition using Non-Information Oracle

In the UC-PAKE ideal functionality $\mathcal{F}_{\text{PAKE}}$, on corruption of a party (if its peer has already issued a session key) the session key is provided to the ideal world adversary. However, it has been pointed out in [CK02] that some efficient protocols, especially two message protocols, cannot realize this ideal functionality under such adaptive corruption, i.e. when a party is corrupted after its peer has emitted the session key. This is the case even though these protocols can be used to build secure channels.

Thus, it is worth considering a relaxed UC-PAKE ideal functionality which can be realized by these efficient protocols. [CK02] suggest using an ideal functionality parameterized by a non-information oracle. A **non-information oracle** $\mathcal{N}$ is an interacting probabilistic Turing machine, which interacts with the ideal world adversary, and outputs a key which is indistinguishable from uniformly random to the ideal world adversary, if the adversary is efficient. In a **multi-session version**, the non-information oracle $\hat{\mathcal{N}}$, starts with a first phase where it receives a message from the ideal world adversary and generates a state σ, and a message τ for the adversary. For example, it can expect a bilinear pairing group along with generators and just check that the generators are of claimed order. In the next phase, for each session it spawns a new $\mathcal{N}$ initialized with σ, and auxiliary information consisting of both passwords stored in records for the two parties in the session (this auxiliary information is provided by the ideal functionality). For each $\mathcal{N}$ we will refer to the random coins, passwords and all other inputs from the adversary used by $\mathcal{N}$ to generate the key from σ as **local witness**. For any party, the adversary can call $\mathcal{N}$ with a corrupt message and $\mathcal{N}$ computes a state ξ from the local witness, the incoming message and password stored in the record for all sessions and this party and gives it to the adversary. We will refer to ξ as **local ample-witness**. On a NewKey call from the adversary, $\mathcal{N}$ computes a key and outputs the key, as its final output. This output is not to the adversary, but to the ideal functionality. On a $\mathsf{NewKeySim}$ call from the adversary, $\mathcal{N}$ computes a key and returns the key to the adversary (i.e. does not output).

For defining non-information multi-session oracle $\hat{\mathcal{N}}$ consider another interactive Turing machine $\hat{\mathcal{M}}$ with the same interface as $\hat{\mathcal{N}}$ such that $\hat{\mathcal{M}}$ *does not* use the passwords provided by the functionality, and further for any adaptively chosen choice of a session by the adversary, on NewKey call it outputs a random and independent key. The adversary cannot call corrupt for this party or the peer in this session if their passwords are the same. $\hat{\mathcal{M}}$ can still use the passwords for computing the local ample-witness. The multi-session oracle $\hat{\mathcal{N}}$ is called a non-information oracle for password-based key-exchange if there exists another interactive Turing machine $\hat{M}$ such that no efficient adversary can distinguish between the interacting with $\hat{\mathcal{N}}$ or $\hat{\mathcal{M}}$.

Thus, in the relaxed (multi-session) ideal functionality $\widehat{\mathcal{F}}^{\hat{\mathcal{N}}}_{\mathrm{RPAKE}}$, the ideal functionality is parameterized by a security parameter k, and a non-information oracle $\hat{\mathcal{N}}$. As a first step in the ideal functionality, the non-information oracle $\hat{\mathcal{N}}$ expects a message from the Adversary, it computes τ and σ, sends τ to the Adversary, and saves σ as the starting state of all the individual session $\mathcal{N}$ it is going to spawn.

Also, on receiving a query $(\mathsf{NewSession}, sid, \mathsf{ssid}, P_i, P_j, pw, role)$ from party P_j, recall the ideal functionality sends $(\mathsf{NewSession}, sid, \mathsf{ssid}, P_i, P_j, role)$ to S. $\hat{\mathcal{N}}$ spawns a new $\mathcal{N}$ for this ssid (unless it was already started) and sets its initial state to σ. Moreover $\mathcal{N}$ is provided with pw as auxiliary information by the ideal functionality. $\mathcal{N}$ can expect the message $(\mathsf{NewSession}, sid, \mathsf{ssid}, P_i, P_j, role)$ from S, which then sends a reply meesage to S.

Further, in the step "in any other case, pick a new random key sk' of length k and send $(sid, \mathsf{ssid}, sk')$ to P_i" is now replaced by "in any other case, send $(\mathsf{NewKey}, sid, \mathsf{ssid}, P_i)$ to S which then calls $\mathcal{N}$ with the same. $\mathcal{N}$ then outputs a key sk' and then the ideal functionality outputs $(sid, \mathsf{ssid}, sk')$ to P_i".

Finally, on corruption of a party P_i, for every session such that there is a record of the form $(\mathsf{ssid}, P_i, P_j, pw)$ which is fresh, and there is a record $(\mathsf{ssid}, P_j, P_i, pw')$ with $pw' = pw$, and a key sk' was sent to P_j in session ssid, and $(\mathsf{ssid}, P_j, P_i, pw)$ was fresh at the time, then output the local coins of $\mathcal{N}$ of that session ssid to S. The password pw is also revealed to S as usual.

6.2.1 Discussion about Non-Information Oracle Relaxation

The ideal functionality $\mathcal{F}^{\mathcal{N}}_{\mathrm{RPAKE}}$ is both a relaxation and a restriction of $\mathcal{F}_{\mathrm{PAKE}}$. It is a restriction because its realization would require $\mathcal{N}$-based protocols. This is not a problem, as we do show such a realization. It is a relaxation because this protocol does not seem to realize $\mathcal{F}_{\mathrm{PAKE}}$. However, this is really an artifact of the $\mathcal{N}$-restriction. The definition of realization says that for every adversary $\mathcal{A}$ in the real world (protocol) its view is indistinguishable from the view of another adversary $\mathcal{S}$ in the ideal world. Since in the ideal world (as long as there is no corruption) the adversary $\mathcal{S}$ gets no information about the session key or the password, it must also be the case in the real world. This property holds for both ideal functionalities $\mathcal{F}_{\mathrm{PAKE}}$ and $\mathcal{F}_{\mathrm{RPAKE}}$. Now, if a party is corrupted after the session key is output by the peer (and it has output its outgoing message to the adversary $\mathcal{A}$), we do expect in the real world for the adversary $\mathcal{A}$ to obtain this session key as well as local state stored at the party to compute the session key (including password). However, the security guarantee should be that the $\mathcal{A}$ should not get anything beyond that. The $\mathcal{F}^{\mathcal{N}}_{\mathrm{RPAKE}}$ ideal functionality discloses to the ideal world adversary $\mathcal{S}$ the session key as well as information it used to generate the session key. This is information which is based completely on the internal random coins of the ideal functionality, and its interaction with adversary, and the passwords. This information is no more than the information we are expected to securely release to the adversary in the real world. In the multi-session version, releasing the local ample-witness of multiple sessions does not affect the security of other sessions as keys in those session continue to appear random to the efficient adversary, by multi-session non-information stipulation. Similarly, information about passwords in the good sessions is zero-knowledge.

Generate $\mathbf{g}_1 \leftarrow \mathbb{G}_1, \mathbf{g}_2 \leftarrow \mathbb{G}_2$ and $a, b, c, d, e, u_1, u_2 \leftarrow \mathbb{Z}_q$, and let $\mathcal{H}$ be a CRHF.
Compute $\mathbf{a} = \mathbf{g}_1^a,\ \mathbf{d} = \mathbf{g}_1^d,\ \mathbf{e} = \mathbf{g}_1^e,\ \mathbf{w}_1 = \mathbf{g}_1^{u_1},\ \mathbf{w}_2 = \mathbf{g}_1^{u_2}$
$\mathbf{b} = \mathbf{g}_2^b,\ \mathbf{c} = \mathbf{g}_2^c,\ \mathbf{v}_1 = \mathbf{g}_2^{u_1 b - d - ca},\ \mathbf{v}_2 = \mathbf{g}_2^{u_2 b - e}$.

$$\text{CRS} := (\mathbf{g}_1, \mathbf{g}_2, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{w}_1, \mathbf{w}_2, \mathbf{v}_1, \mathbf{v}_2, \mathcal{H}).$$

Party P_i	Network
Input (`NewSession`, sid, ssid, P_i, P_j, pwd, $initiator/responder$) Choose $r_1, s_1 \xleftarrow{\$} \mathbb{Z}_q$. Set $R_1 = \mathbf{g}_1^{r_1}$, $S_1 = \text{pwd} \cdot \mathbf{a}^{r_1}$, $T_1 = (\mathbf{d} \cdot \mathbf{e}^{i_1})^{r_1}$, $\hat{\rho}_1 = \mathbf{b}^{s_1}$, $\rho_1 = \mathbf{g}_2^{s_1}$, $\theta_1 = \mathbf{c}^{s_1}$, $\gamma_1 = (\mathbf{v}_1 \mathbf{v}_2^{i'_2})^{s_1}$. $W_1 = (\mathbf{w}_1 \mathbf{w}_2^{i_1})^{r_1}$, where $i_1 = \mathcal{H}(sid, \mathsf{ssid}, P_i, P_j, R_1, S_1, \hat{\rho}_1)$ and erase r_1, s_1. Send $R_1, S_1, T_1, \hat{\rho}_1$, and retain $\rho_1, \theta_1, \gamma_1, W_1$.	$\xrightarrow{R_1, S_1, T_1, \hat{\rho}_1} P_j$
Receive $R'_2, S'_2, T'_2, \hat{\rho}'_2$. Let $i'_2 = \mathcal{H}(sid, \mathsf{ssid}, P_j, P_i, R'_2, S'_2, \hat{\rho}'_2)$ If any of $R'_2, S'_2, T'_2, \hat{\rho}'_2$ is not in their respective group or is 1, set $\text{sk}_1 \xleftarrow{\$} \mathbb{G}_T$, else Compute $\text{sk}_1 = e(T'_2, \rho_1) \cdot e(S'_2/\text{pwd}, \theta_1) \cdot e(R'_2, \gamma_1) \cdot e(W_1, \hat{\rho}'_2)$ Output $(sid, \mathsf{ssid}, \text{sk}_1)$.	$\xleftarrow{R'_2, S'_2, T'_2, \hat{\rho}'_2} P_j$

Figure 3: Single round UC-secure Password-authenticated KE under SXDH Assumption.

6.3 Main Idea of the UC Protocol using DSS-QA-NIZK

For the sake of exposition, let's call one party in the session the server and the other the client. (There is no such distinction in the actual protocol, and in fact each party will run two parallel protocols, one as a client and another as a server, and output the product of the two keys generated). The common reference string (CRS) defines a Diffie-Hellman language, i.e. $\rho = \mathbf{g}_1, \mathbf{g}_1^a$. The client picks a fresh Diffie-Hellman tuple by picking a witness r and computing $\langle \mathbf{x}_1 = \mathbf{g}_1^r, \mathbf{x}_2 = \mathbf{g}_1^{a \cdot r} \rangle$. It also computes a DSS-QA-NIZK proof on this tuple, which is a hash proof T and a QA-NIZK proof W of the augmented Diffie-Hellman tuple. Note, the QA-NIZK proof W is just a single group element [JR14] (see Appendix B.2 for details). It next modifies the Diffie-Hellman tuple using the password pwd it possesses. Essentially, it multiplies $\mathbf{x}_2$ by pwd to get a modified group element which we will denote by S. It next sends this modified Diffie-Hellman tuple, i.e. $\mathbf{x}_1, S$, and the T component of the proof to the server. It retains W for later use. At this point it can erase the witness r.

As a first step, we intend to utilize an interesting property of the real-world verifier V of the DSS-QA-NIZK: the verifier is just the verifier of the QA-NIZK for the DH language augmented with the hash proof, and the QA-NIZK verifiers for linear subspaces are just a single bi-linear product test. Specifically (see Appendix B.2), V on input $\mathbf{x}_1, \mathbf{x}_2$ and proof T, W, computes $\iota = \mathcal{H}(\mathbf{x}_1, \mathbf{x}_2)$, and outputs true iff

$$e(\mathbf{x}_1, (\mathbf{v}_1 \mathbf{v}_2^{\iota})) \cdot e(\mathbf{x}_2, \mathbf{c}) \cdot e(T, \mathbf{g}_2) \; = \; e(W, \mathbf{b}).$$

Thus, it outputs true iff the left-hand-size (LHS) equals the right-hand-side (RHS) of the above equation. Note that the client sent $\mathbf{x}_1, S$ (i.e. $\mathbf{x}_2$ linearly modified by pwd) and T to the server. Assuming the server has the same password pwd, it can un-modify the received message and get $\mathbf{x}_2 = S/\text{pwd}$, and hence can compute this LHS (using the CRS). The client retained W, and can compute the RHS (using the CRS).

The intuition is that unless an adversary out-right guesses the password, it cannot produce a different $\mathbf{x}_1'$, S', T', such that $\mathbf{x}_1'$, S'/pwd, T' used to compute the LHS will match the RHS above. While we make this intuition rigorous later by showing a UC simulator, to complete the description of the protocol, and using this intuition, the client and server actually compute the LHS and RHS respectively of the following equation (for a fresh random $s \in \mathbb{Z}_q$ picked by the server):

$$e(\mathbf{x}_1, (\mathbf{v}_1\mathbf{v}_2^{\iota})^s) \cdot e(\mathbf{x}_2, \mathbf{c}^s) \cdot e(T, \mathbf{g}_2^s) \;=\; e(W, \mathbf{b}^s). \tag{1}$$

Now note that for the client to be able to compute the RHS, it must have $\mathbf{b}^s$, since s was picked by the server afresh. For this purpose, the protocol requires that the server send $\mathbf{b}^s$ to the client (note this can be done independently and asynchronously of the message coming from the client). It is not difficult to see, from completeness of the prover and verifier of the DSS-QA-NIZK, that both parties compute the same quantity.

As mentioned earlier, each pair of parties actually run two versions of the above protocol, wherein each party plays the part of client in one version, and the part of server in the other version. Each party then outputs the product of the LHS of (1) computation (in the server version) and the RHS of (1) computation (in the client version) as the session-key. We will refer to these two factors in the session-key computation as the *server factor* and the client factor resp. This is the final UC-PAKE protocol described in Fig. 6.2.1 (with the parties identities, session identifiers and $\mathbf{b}^s$ from its server version, used as label). The quantity $\mathbf{x}_1$ is called R in the protocol, as subscripts will be used for other purposes.

Theorem 9 *Assuming the existence of SXDH-hard groups, the protocol in Fig 6.2.1 securely realizes the $\widehat{\mathcal{F}}_{\text{PAKE}}$ functionality in the $\mathcal{F}_{\text{CRS}}$ hybrid model, in the presence of adaptive corruption adversaries, as long as an adversary does not corrupt a party after its peer has issued a key in a session and the adversary has not delivered the message in the session from the peer to the party.*

The theorem is proved in Appendix D. We provide the intuition below.

Theorem 10 *Assuming the existence of SXDH-hard groups, there exists a multi-session non-information oracle $\hat{\mathcal{N}}$ such that the protocol in Fig 6.2.1 securely realizes the $\widehat{\mathcal{F}}^{\hat{\mathcal{N}}}_{\text{RPAKE}}$ functionality in the $\mathcal{F}_{\text{CRS}}$ hybrid model, in the presence of adaptive corruption adversaries.*

In appendix E we describe the changes required in the proof of theorem 9 to get the proof of this theorem.

6.4 Main Idea of the UC Simulator

We first *re-define* the various verifiers in the DSS-QA-NIZK for the DH language described in Section B.2, to bring them in line with the above description. In particular, the real-world verifier V is defined equivalently to be: the verifier V takes as input $\mathbf{CRS}_v$, a potential language member $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle$, and a proof $\pi = (T, W)$, computes $\iota = \mathcal{H}(\mathbf{x}_1, \mathbf{x}_2, l)$, picks a fresh random $s \in \mathbb{Z}_q$, and outputs true iff

$$e(\mathbf{x}_1, (\mathbf{v}_1\mathbf{v}_2^{\iota}))^s \cdot e(\mathbf{x}_2, \mathbf{c})^s \cdot e(T, \mathbf{g}_2)^s \;=\; e(W, \mathbf{b}^s).$$

This is equivalent as long as $s \neq 0$.

The partial-simulation world private-verifier pV is now defined as: it checks a potential language member $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle$ and a proof T, W as follows: compute $\iota = \mathcal{H}(\mathbf{x}_1, \mathbf{x}_2, l)$; pick s and s' randomly and independently from $\mathbb{Z}_q$, and if $\mathbf{x}_2 = \mathbf{x}_1^a$ and $T = \mathbf{x}_1^{d+\iota e}$ then set $\xi = \mathbf{1}_T$ else set $\xi = e(\mathbf{g}_1, \mathbf{g}_2)^{s'}$ and output true iff

$$e(\mathbf{x}_1, (\mathbf{v}_1\mathbf{v}_2^{\iota}))^s \cdot e(\mathbf{x}_2, \mathbf{c})^s \cdot e(T, \mathbf{g}_2)^s \cdot \xi \;=\; e(W, \mathbf{b}^s). \tag{2}$$

This is equivalent to the earlier definition of pV with high probability by an information-theoretic argument, if the trapdoors used were generated by the semi-functional CRS generator sfK_1.

The UC simulator $\mathcal{S}$ works as follows: It will generate the CRS for $\widehat{\mathcal{F}}_{\text{PAKE}}$ using the semi-functional CRS generator sfK_1 for the Diffie-Hellman language. The next main difference is in the simulation of the outgoing message of the real world parties: $\mathcal{S}$ uses a dummy message μ instead of the real password which it does not have access to. Further, it postpones computation of W till the session-key generation time. Finally, another difference is in the processing of the incoming message, where $\mathcal{S}$ decrypts the incoming message R_2', S_2', T_2' to compute a pwd$'$, which it uses to call the ideal functionality's test function. It next generates a sk similar to how it is generated in the real-world (recall the computation of server factor and client factor by LHS and RHS of (1)) except that it uses the equation (2) corresponding to the private verifier. It sends sk to the ideal functionality to be output to the party concerned.

Note, $\mathcal{S}$ simulating the server factor computation can compute the LHS of equation (2), except $\mathcal{S}$ does not have direct access to pwd and hence cannot get $\mathbf{x}_2$ from the modified $\hat{S}$ that it receives. However, it can do the following: Use the $\mathsf{TestPwd}$ functionality of the ideal functionality $\widehat{\mathcal{F}}_{\text{PAKE}}$ with a pwd$'$ computed as $\hat{S}/\mathbf{x}_1^a$. If this pwd$'$ does not match the pwd recorded in $\widehat{\mathcal{F}}_{\text{PAKE}}$ for this session and party, then $\widehat{\mathcal{F}}_{\text{PAKE}}$ anyway outputs a fresh random session key, which will then turn out to be correct simulation (note, this case is same as $\mathbf{x}_2$ $(= S/\text{pwd})$ $\neq \mathbf{x}_1^a$, which would also have resulted in the same computation on the LHS). If the pwd$'$ matched the pwd, the simulator is notified the same, and hence it can now do the following: if $T = \mathbf{x}_1^{d+\iota e}$ then set $\xi = \mathbf{1}_T$ else set $\xi = \epsilon(\mathbf{g}_1, \mathbf{g}_2)^{s'}$. Next, it calls $\widehat{\mathcal{F}}_{\text{PAKE}}$'s NewKey with session key $e(\mathbf{x}_1, (\mathbf{v}_1\mathbf{v}_2^{\iota}))^s \cdot e(\mathbf{x}_1^a, \mathbf{c})^s \cdot e(T, \mathbf{g}_2)^s \cdot \xi$ (multiplied by a RHS computation of (2) in simulation of the client factor, which we will discuss later).

The UC Simulator $\mathcal{S}$ must also simulate $\mathbf{g}_1^r, \text{pwd} \cdot (\mathbf{g}_1^a)^r$ and the T component of the DSS-QA-NIZK, as that is the message sent out to the adversary by the real party ("client" part of the protocol). However, $\mathcal{S}$ does not have access to pwd. It can just generate a fake tuple $\mathbf{g}_1^r, \mu \cdot (\mathbf{g}_1^a)^r \cdot \mathbf{g}_1^{r'}$ (for some constant or randomly chosen group element μ, and some random and independent $r' \in \mathbb{Z}_q$). Now, the semi-functional (proof) simulator sfSim of the DSS-QA-NIZK of Section B.2 has an interesting property that when the tuple $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle$ does not belong to the language (language membership-bit zero), the T component of the simulated proof can just be generated randomly.

The simulator also needs W to compute the client factor, and we had postponed it till the session-key computation phase. As mentioned above, if the password pwd$'$ "decrypted" from the incoming message is not correct then the key is anyway set to be random, and hence a proper W is not even required. However, if the pwd$'$ is correct, the simulator is notified of same, and hence it can compute W component of the proof by passing $\mathbf{x}_2 = \mu \cdot (\mathbf{g}_1^a)^r \cdot \mathbf{g}_1^{r'}/\text{pwd}'$ along with $\mathbf{x}_1$ $(= \mathbf{g}_1^r)$ to sfSim.

Of course, fixing the above fake tuples employs one-time full-simulation property of the DSS-QA-NIZK (and the DDH assumption).

6.5 Main Idea of the Proof of UC Realization

The proof that the simulator $\mathcal{S}$ described above simulates the Adversary in the real-world protocol, follows essentially from the properties of the DSS-QA-NIZK, although not generically since the real-world protocol and the simulator use the verifiers V and pV (resp.) in a split fashion. However, as described above the proof is very similar and we give a broad outline here. The proof will describe various experiments between a challenger $\mathcal{C}$ and the adversary, which we will just assume to be the environment $\mathcal{Z}$ (as the adversary $\mathcal{A}$ can be assumed to be just dummy and following $\mathcal{Z}$'s commands). In the first experiment the challenger $\mathcal{C}$ will just be the combination of the code of the simulator $\mathcal{S}$ above and $\widehat{\mathcal{F}}_{\text{PAKE}}$. In particular, after the environment issues a $\mathsf{NewSession}$ request with a password pwd, the challenger gets that password. So, while in the first experiment, the challenger (copying $\mathcal{S}$) does not use pwd directly, from the next experiment on-wards, it can use the pwd. Thus, the main goal of the ensuing experiments is to modify the fake tuples $\mathbf{g}_1^r, \mu \cdot (\mathbf{g}_1^a)^r \cdot \mathbf{g}_1^{r'}$ by real tuples (as in real-world) $\mathbf{g}_1^r, \text{pwd} \cdot (\mathbf{g}_1^a)^r$, since the challenger has access to pwd. This is accomplished by a hybrid argument, modifying one instance at a time using DDH assumption in group $\mathbb{G}_1$ and using one-time full-ZK property (and using the otfSim proof simulator for that instance). A variant of the one-time full-ZK semi-functional verifier sfV (just as the variants for pV and V described above) is easily obtained. Note that in each experiment, whenever the simulator invokes partial proof simulation it can provide the correct membership bit (with high probability) as in each experiment it knows exactly which tuples are real and which are fake.

Once all the instances are corrected, i.e. R, S generated as $\mathbf{g}_1^r, \text{pwd} \cdot (\mathbf{g}_1^a)^r$, the challenger can switch to the real-world because the tuples $R, S/\text{pwd}$ are now Diffie-Hellman tuples. This implies that the session keys are generated using the V variant described above, which is exactly as in the real-world.

6.6 Adaptive Corruption

The UC protocol described above is also UC-secure against adaptive corruption of parties by the Adversary in the erasure model. In the real-world when the adversary corrupts a party (with a $\mathsf{Corrupt}$ command), it gets the internal state of the party. Clearly, if the party has already been invoked with a $\mathsf{NewSession}$ command then the password pwd is leaked at the minimum, and hence the ideal functionality $\mathcal{F}_{\text{PAKE}}$ leaks the password to the Adversary in the ideal world. In the protocol described above, the Adversary also gets W and s, as this is the only state maintained by each party between sending $R, S, T, \hat{\rho}$, and the final issuance of session-key. Simulation of s is easy for the simulator $\mathcal{S}$ since $\mathcal{S}$ generates s exactly as in the real world. For generating W, which $\mathcal{S}$ had postponed to computing till it received an incoming message from the adversary, it can now use the pwd which it gets from $\widehat{\mathcal{F}}_{\text{PAKE}}$ by issuing a $\mathsf{Corrupt}$ call to $\widehat{\mathcal{F}}_{\text{PAKE}}$. More precisely, it issues the $\mathsf{Corrupt}$ call, and gets pwd, and then calls the semi-functional simulator with $\mathbf{x}_2 = \mu \cdot (\mathbf{g}_1^a)^r \cdot \mathbf{g}_1^{r'}/\text{pwd}$ along with $\mathbf{x}_1$ $(= \mathbf{g}_1^r)$ to get W. Note that this computation of W is identical to the postponed computation of W in the computation of client factor of sk_1 (which is really used in the output to the environment when $\text{pwd}' = \text{pwd}$).

References

[ABB+13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. In Kazue Sako and Palash

Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 214–234. Springer, Heidelberg, December 2013. (document), 1.0.1, 2

[ABP15] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Disjunctions for hash proof systems: New constructions and applications. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 69–100. Springer, Heidelberg, April 2015. 1, 1

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 671–689. Springer, Heidelberg, August 2009. 1.0.1, 2

[BBC$^+$13] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 449–475. Springer, Heidelberg, August 2013. 1.0.1, 2

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. 6

[CCS09] Jan Camenisch, Nishanth Chandran, and Victor Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 351–368. Springer, Heidelberg, April 2009. 1, 1.0.1

[CHK$^+$05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, Heidelberg, May 2005. 1.0.1, 6

[CK02] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 337–351. Springer, Heidelberg, April / May 2002. 1.0.1, 6.2

[CR03] Ran Canetti and Tal Rabin. Universal composition with joint state. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 265–281. Springer, Heidelberg, August 2003. 6.1

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, April / May 2002. 1, 4, C.1, C.1

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *Proc. 23rd ACM STOC*, pages 542–552, 1991. 1

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 15

[EHO$^+$13] Keita Emura, Goichiro Hanaoka, Go Ohtake, Takahiro Matsuda, and Shota Yamada. Chosen ciphertext secure keyed-homomorphic public-key encryption. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 32–50. Springer, Heidelberg, February / March 2013. 1.0.1, 5

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

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008. 1, 1, 5

[Har11] Kristiyan Haralambiev. Efficient cryptographic primitives for non-interactive zero-knowledge proofs and applications. *PhD Dissertation*, 2011. 3, 5

[JR12] Charanjit S. Jutla and Arnab Roy. Relatively-sound NIZKs and password-based key-exchange. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 485–503. Springer, Heidelberg, May 2012. 1.0.1, 2, 3, 5

[JR13] Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2013. 1, 1, 1.0.1, 2, 1, F, F

[JR14] Charanjit S. Jutla and Arnab Roy. Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 295–312. Springer, Heidelberg, August 2014. 1, 4, 4, 6.3, B.1, B.1, B.2.1, F

[KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 293–310. Springer, Heidelberg, March 2011. 1.0.1, 2

[KW15] Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 101–128. Springer, Heidelberg, April 2015. 1, 1

[LPJY14] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Non-malleability from malleability: Simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 514–532. Springer, Heidelberg, May 2014. 1, 1.0.1, 5

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, Heidelberg, August 2002. 1.0.1

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999. 1

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, August 2009. 1, 1.0.1, F

A Proofs of Dual-System Lemmas

Lemma 4 *(true-simulation-soundness)* For a DSS-QA-NIZK, for all PPT adversaries $\mathcal{A}$,
$\Pr[(\rho, \sigma, \tau, \eta) \leftarrow \mathsf{sfK}_1(\lambda);\ (x, l, \pi) \leftarrow \mathcal{A}^{\mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot)}\ (\sigma, \rho)\ :\ (x \notin L_\rho)\ \wedge\ \mathsf{V}(\sigma, x, l, \pi) = 1]\ \approx 0$,
where the experiment aborts if $\mathcal{A}$ invokes the oracle with some (y, β, l) , s.t. $y \notin L_\rho$ or $\beta = 0$.

Let ADV^{pzk} be the maximum (absolute value of) difference in probability (over all Adversaries) in the composable partial-ZK property of the DSS-QA-NIZK, when the Adversary makes at most one call to the third oracle.
Then, the maximum success probability above is $\text{ADV}^{\text{pzk}} + \text{ADV}^{\text{pss}}$.
Proof: Let the above probability be Δ. Then $\mathcal{A}$ produces a tuple (x, l, π), such that $x \notin L_\rho$ with probability $\Delta_1 \geq \Delta$. Let Δ_2 be the probability that $\mathcal{A}$ produces a tuple with $x \in L_\rho$ and yet V rejects. Finally, let Δ_3 be the probability that $\mathcal{A}$ produces a tuple with $x \in L_\rho$ and V accepts. Thus, $\Delta_2 = 1-(\Delta_1+\Delta_2)$. Moreover, the probability that V accepts is $\Delta+\Delta_3$. Now, by unbounded partial-simulation soundness, the probability that $\mathcal{A}$ produces a tuple with $x \notin L_\rho$ or V rejects, and yet pV accepts the tuple is negligible. Thus, on $\mathcal{A}$'s output, pV rejects with probability at least $\Delta_1 + \Delta_2 - \epsilon$, where ϵ is negligible. Now we build an adversary $\mathcal{A}_1$, which emulates $\mathcal{A}$ and instead of outputting the final tuple, calls an oracle with that tuple, and outputs the value returned by the oracle. If the oracle is V, then $\mathcal{A}_1$ outputs 0 with probability $(\Delta_1 - \Delta) + \Delta_2$. If the oracle is pV, $\mathcal{A}_1$ outputs 0 with probability at least $\Delta_1 + \Delta_2 - \epsilon$. Then, by partial-ZK (second part) property, Δ must be negligible. □ □

Lemma 11 *For a DSS-QA-NIZK for a parameterized language, for distributions $\mathcal{U}$ of fake language members (i.e. $\beta = 0$) such that with high probability no efficient adversary can distinguish them from another witness-sampleable distribution $\mathcal{E}$ of real language members (i.e. $\beta = 1$), the semi-functional proofs generated on such tuples by* sfSim *are accepted by* V.

Proof: Let the probability in the following experiment be Δ.

$$\Pr[(\rho, \sigma, \tau, \eta) \leftarrow \mathsf{sfK}_1(\lambda);\ x \leftarrow \mathcal{U}; \pi \leftarrow \mathsf{sfSim}(\sigma, \tau, x, \beta = 0); \mathsf{V}(\sigma, x, \pi) = 1] = \Delta.$$

Then, since V is a polynomial time algorithm, and that does not need any trapdoors, it follows by one-time full simulation, that the following probability is close to Δ.

$$\Pr[\rho \leftarrow \mathcal{D}_\lambda; (\sigma, \tau, \tau_1\eta) \leftarrow \mathsf{otfK}_1(\lambda, \rho);\ x \leftarrow \mathcal{U}; \pi \leftarrow \mathsf{otfSim}(\sigma, \tau_1, x); \mathsf{V}(\sigma, x, \pi) = 1] \approx \Delta.$$

Note V is not being used as an oracle. Next, the following probability is also close to Δ, by computational indistinguishability of $\mathcal{E}$ and $\mathcal{U}$, and noting that otfK_1, otfSim and V are PPT, and $x \leftarrow \mathcal{U}$ (or $x \leftarrow \mathcal{E}$) can be sampled before otfK_1 is invoked, and $\mathcal{E}$ is witness sampleable.

$$\Pr[\rho \leftarrow \mathcal{D}_\lambda; (\sigma, \tau, \tau_1\eta) \leftarrow \mathsf{otfK}_1(\lambda, \rho);\ (x, w) \leftarrow \mathcal{E}; \pi \leftarrow \mathsf{otfSim}(\sigma, \tau_1, x); \mathsf{V}(\sigma, x, \pi) = 1]$$
$$\approx \Delta.$$

Again, by one-time full simulation we get,

$$\Pr[(\rho, \sigma, \tau, \eta) \leftarrow \mathsf{sfK}_1(\lambda);\ (x, w) \leftarrow \mathcal{E}; \pi \leftarrow \mathsf{sfSim}(\sigma, \tau, x, \beta = 1); \mathsf{V}(\sigma, x, \pi) = 1] \approx \Delta.$$

Now, by composable partial-simulation (second part), we get

$$\Pr[(\rho, \sigma, \tau, \eta) \leftarrow \mathsf{sfK}_1(\lambda);\ (x, w) \leftarrow \mathcal{E}; \pi \leftarrow \mathsf{P}(\sigma, x, w); \mathsf{V}(\sigma, x, \pi) = 1] \approx \Delta.$$

Now, by composable partial-simulation (first part), we get

$$\Pr[\rho \leftarrow \mathcal{D}_\lambda; \sigma \leftarrow \mathsf{K}_1(\lambda);\ (x, w) \leftarrow \mathcal{E}; \pi \leftarrow \mathsf{P}(\sigma, x, w); \mathsf{V}(\sigma, x, \pi) = 1] \approx \Delta.$$

But, by completeness this probability is close to one, and hence Δ is close to one. □ □

Lemma 12 (simulation-soundness of semi-functional verifier) *If the one-time full-simulation property holds statistically, then in the one-time simulation world, the semi-functional verifier* sfV *is sound for all inputs except the tuple corresponding to the one-time full-simulation. In other words, for all PPT adversaries* $\mathcal{A}_3, \mathcal{A}_4$,

$$\begin{aligned}
&\Pr[\rho \leftarrow \mathcal{D}_\lambda;\ (\sigma, \tau, \tau_1, \eta)\ \leftarrow\ \mathsf{otfK}_1(\lambda, \rho);\\
&\quad (x^*, l^*, \beta^*, s) \leftarrow \mathcal{A}_3^{\mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot),\, \mathsf{sfV}(\sigma,\eta,\cdot,\cdot,\cdot)}(\sigma, \rho);\\
&\quad \pi^* \leftarrow \mathsf{otfSim}(\sigma, \tau_1, x^*, l^*); (x, l, \pi) \leftarrow \mathcal{A}_4^{\mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot),\, \mathsf{sfV}(\sigma,\eta,\cdot,\cdot,\cdot)}(\pi^*, s) :\\
&\quad (x \notin L_\rho)\ \wedge \mathsf{sfV}(\sigma, \eta, x, l, \pi) = 1] \approx 0,
\end{aligned}$$

where the experiment aborts if either in the call to the first oracle, or in the (x^*, β^*) *produced by* $\mathcal{A}_3$*, the membership-bit provided is not the correct* L_ρ*-membership-bit,* **or** *if* $\langle x^*, l^*, \pi^* \rangle$ *is queried to* sfV/pV*, or is same as* $\mathcal{A}_4$*'s output.*

Let ADV^{pss} be the maximum success probability (over all Adversaries) in the partial-simulation soundness experiment of the DSS-QA-NIZK.
Let $\text{DIST}^{\text{otzk}}$ be the maximum statistical distance between the views of the adversaries $(\mathcal{A}_3, \mathcal{A}_4)$ in the one-time full-ZK property of DSS-QA-NIZK Σ.
Then, the maximum success probability above is at most $\text{ADV}^{\text{pss}} + \text{DIST}^{\text{otzk}}$.
Proof: Let the probability of the above event be Δ. Consider an adversary $\mathcal{A}_5$ which is same as $\mathcal{A}_4$ except that instead of outputting (x, l, π), it calls sfV on that same tuple, and returns (γ, x, l, π) where γ is the bit returned by this final sfV call. Thus, the probability that $\mathcal{A}_5$ outputs $(1, x, l, \pi)$ with $x \notin L_\rho$ is Δ. Since, the one-time full-simulation property holds statistically, the view of $\mathcal{A}_3, \mathcal{A}_5$ is (almost) identical in the partial-simulation world (i.e. with the one-time full-simulation also replaced by sfSim and sfV replaced by pV). In particular, the probability that $\mathcal{A}_5$ outputs $(1, x, l, \pi)$ with $x \notin L_\rho$ is negligibly close to Δ. But, the first component of $\mathcal{A}_5$'s output equals pV invoked on (x, l, π), and hence pV outputs 1 on (x, l, π) produced by PPT adversary $\mathcal{A}_5$, with $x \notin L_\rho$, with probability close to Δ. But, by unbounded partial-simulation soundness this probability is negligible, and hence Δ is negligible. □ □

B Proof of General DSS-QA-NIZK Construction

In this appendix, we prove theorem 6 about the general DSS-QA-NIZK construction of Section 4. Let

- $\text{ADV}^{\text{sound}}(Q)$ be the maximum success probability of any adversary in the soundness experiment of QA-NIZK Q.

- $\text{ADV}^{\text{wss}}(Q)$ be the maximum success probability of any adversary in the true-simulation-soundness experiment of QA-NIZK Q (see Definition 5).

- $\text{ADV}^{\text{zk}}(Q)$ be the maximum (absolute value of) difference in probability (over all Adversaries) in the composable-ZK property of QA-NIZK Q.

- $\text{ADV}^{\text{comp}}(Q)$ be the maximum success probability of any adversary in the completeness experiment of QA-NIZK Q. If the probability of completeness holding is $1-y$, then the Adversary's success probability is considered to be y.

- Let ϵ_1 be the failure probability of algorithm E_2.

- δ be the maximum probability that the simulator of Q does not generate unique acceptable proofs (see Definition 5).

Recall ϵ is the approximation probability in the smoothness and universal$_2$ properties of hash function H.
We start with two simple technical lemmas.

Lemma 13 (Strong Composability) *In the context of Theorem 6, for DSS-QA-NIZK Σ, for every PPT adversary $\mathcal{B}$ (with access to trapdoor τ),*
$\Pr[(\rho, \sigma, \tau, \eta) \leftarrow \mathsf{sfK}_1(\lambda) : \mathcal{B}^{\mathsf{P}(\sigma,\cdot,\cdot,\cdot)}(\sigma, \rho, \tau) = 1] \approx$
$\Pr[(\rho, \sigma, \tau, \eta) \leftarrow \mathsf{sfK}_1(\lambda) : \mathcal{B}^{\mathsf{sfSim}^*(\sigma,\tau,\cdot,\cdot,\cdot)}(\sigma, \rho, \tau) = 1]$,
where the experiment aborts if $\mathcal{B}$ calls the oracle with some (x, w, l) s.t. $(w, x) \notin R_\rho$.

The proof of the above lemma follows easily from the description of the construction, and the facts that H is a projective hash function and QA-NIZK Q for $L^*_{\rho,\alpha(k)}$ has composable ZK.
Proof: Suppose to the contrary a PPT adversary $\mathcal{B}$ can distinguish the two worlds above. Then, the composable-ZK property of Q does not hold, as we show by building an adversary $\mathcal{S}$ that breaks this property: given the CRS σ' and trapdoor trap (generated by $\mathsf{crs\text{-}sim}$ of Q), $\mathcal{S}$ samples k from K_λ, and outputs $(\sigma', \alpha(k))$ and (k, trap) as CRS σ and simulation trapdoor τ to $\mathcal{B}$. By construction of the DSS-QA-NIZK, this exactly simulates both the experiments for $\mathcal{B}$ so far. The oracles of $\mathcal{B}$, i.e. P and sfSim in the two worlds are also simulated easily using the oracle for $\mathcal{S}$ (i.e. prover and sim resp.); $\mathcal{B}$ supplies an (x, w, l), with $(x, w) \in R_\rho$, from which $\mathcal{S}$ calculates the T value using H_k, and hence $(x, T) \in L^*_{\rho,\alpha(k)}$. $\mathcal{S}$ calls its oracle (which is either prover or sim resp.) with (x, T, l) and witness w and gets as a response a proof W. The oracle of $\mathcal{B}$ is simulated by replying on input (x, w, l) with (T, W). Since x is in L_ρ, the actual oracles of $\mathcal{B}$ are simulated perfectly. Finally, $\mathcal{S}$ outputs a bit exactly as $\mathcal{B}$ does, and hence if $\mathcal{B}$ manages to distinguish the two worlds, then $\mathcal{S}$ manages to break the composable ZK property of Q. $\square$ $\square$

The difference in the probabilities above is at most $\text{ADV}^{\text{zk}}(Q)$.

Lemma 14 (true-simulation-soundness of V) *Recall, in the DSS-QA-NIZK Σ, the CRS σ is of the form $(\sigma', \alpha(k))$, and a proof π is of the form (T, W). In this context, for every PPT adversary $\mathcal{B}$,*
$\Pr[(\rho, \sigma, \tau, \eta) \leftarrow \mathsf{sfK}_1(\lambda); (x, l, \pi) \leftarrow \mathcal{B}^{\mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot)}(\sigma, \rho) = 1\text{: } (x \notin L^*_{\rho,\alpha(k)}) \wedge \mathsf{V}(\sigma, x, l, \pi) = 1] \approx 0$,
where the experiment aborts if $\mathcal{B}$ calls the oracle with some (x', β', l') with $x' \notin L_\rho$, or $\beta' \neq 1$.

The proof follows easily from the fact that in the construction, the QA-NIZK Q is true-simulation-sound.

Proof: Suppose there is an adversary $\mathcal{B}$ that has success probability Δ in the above experiment. We will show a PPT adversary $\mathcal{S}$ that has success probability at least Δ in the true-simulation-sound experiment of QA-NIZK Q. Given the CRS σ' (generated by $\mathsf{crs\text{-}sim}$ of Q), $\mathcal{S}$ samples k from K_λ, and outputs $(\sigma', \alpha(k))$ as CRS σ to $\mathcal{B}$. Adversary $\mathcal{S}$ retains k. The oracle of $\mathcal{B}$ is simulated by $\mathcal{S}$ as follows: On input (x', β', l') with $x' \in L_\rho$ (note β' is required to be 1 as well), $\mathcal{S}$ computes $T' = H_k(x')$, and calls its own oracle sim with (x', T', l') (which is in the augmented language $L^*_{\rho,\alpha(k)}$). This oracle returns a proof W', and $\mathcal{S}$ replies to the oracle call of B with (T', W'). This is exactly the same response of sfSim when membership-bit is 1, and hence simulation of the oracle is perfect. Now note that V of DSS-QA-NIZK behaves as follows: on input CRS $= (\sigma', s)$, x, l, and proof (T, W), it returns the value $\mathsf{ver}(\sigma', \langle x, T, l\rangle, W)$ (using ver of Q). Thus, if $\mathcal{B}$ outputs $x, l, (T, W)$, adversary $\mathcal{S}$ just outputs $\langle x, T, l\rangle$ and proof W. By assumption, the probability of $(x \notin L^*_{\rho,\alpha(k)})$ and $\mathsf{V}((\sigma', \alpha(k)), x, l, (T, W)) = 1$ is Δ. In that event, it follows trivially that ver of Q outputs 1 on $\mathcal{S}$'s output. and $\langle x, T, l\rangle \notin L^*_{\rho,\alpha(k)}$. Thus, success probability of $\mathcal{S}$ is at least Δ. □□

The probability of success above is at most $\textsc{adv}^{\text{wss}}(Q)$.
Let

- N_1, N_2, and N_3 be the number of oracle calls (to the first, second and third oracle resp.) by $\mathcal{A}_1$ in the composable partial-ZK property.

- L_1, L_2 be the number of oracle calls (to the first and second oracle resp.) by $\mathcal{A}_2$ in the partial simulation-soundness experiment.

- M_1, M_2 be the number of calls to the first oracle and second oracle resp. by $\mathcal{A}_3$ and $\mathcal{A}_4$ combined in the one-time full-ZK experiments.

Also, let

- $\textsc{adv}^{\text{comp}}(\Sigma)$ be the maximum success probability of any adversary in the completeness experiment of DSS-QA-NIZK Σ. If the probability of completeness holding is $1-y$, then the Adversary's success probability is considered to be y.

- $\textsc{adv}^{\text{sound}}(\Sigma)$ be the maximum success probability of any adversary in the soundness experiment of DSS-QA-NIZK Σ.

- $\textsc{adv}^{\text{pzk}}(\Sigma, N_3)$ be the maximum (absolute value of) difference in probability (over all Adversaries) in the composable partial-ZK property of DSS-QA-NIZK Σ, with N_3 calls to the third oracle.

- $\textsc{adv}^{\text{pss}}(\Sigma)$ be the maximum success probability of any adversary in the partial-simulation soundness experiment of DSS-QA-NIZK Σ.

- $\textsc{dist}^{\text{otzk}}(\Sigma)$ be the maximum statistical distance (over all Adversaries) between the views of the adversaries $(\mathcal{A}_3, \mathcal{A}_4)$ in the one-time full-ZK property of DSS-QA-NIZK Σ.

In the following proof of Theorem 6, we also show that

- $\textsc{adv}^{\text{comp}}(\Sigma) \leq \textsc{adv}^{\text{comp}}(Q)$,

- $\text{ADV}^{\text{sound}}(\Sigma) \leq \text{ADV}^{\text{sound}}(Q)$,
- $\text{ADV}^{\text{pzk}}(\Sigma, N_3) \leq N_3 \cdot \text{ADV}^{\text{wss}}(Q) + \text{ADV}^{\text{zk}}(Q)$,
- $\text{ADV}^{\text{pss}}(\Sigma) \leq L_2 * \epsilon_1$,
- $\text{DIST}^{\text{otzk}}(\Sigma) \leq (\epsilon + \delta) * (1 + M_2)$.

Proof: (Theorem 6)

Completeness: The verifier V is complete w.r.t. prover P, because if x is in L_ρ with witness w, then the prover P generates proof as (T, W) where $T = \hat{H}(s, x, w)$ and $W = \mathsf{prover}(\sigma, \langle x, T, l\rangle, w)$. Thus, $\langle x, T, l\rangle$ is in $L^*_{\rho,\alpha(k)}$ with witness w, because the public evaluation algorithm $\hat{H}$ computes $H_k(x, l)$ correctly when $x \in L_\rho$. Completeness then follows by noting that verifier V is just the verifier ver of Q, and by completeness of ver of Q w.r.t. prover prover of Q. If Q has perfect completeness, then the DSS-QA-NIZK also has perfect completeness.

Soundness: Let the adversary generate a potential L_ρ member x, a proof (T, W) and a label l such that $x \notin L_\rho$. We now show that the verifier V rejects with high probability. Indeed, V is same as verifier ver of Q, and clearly $\langle x, T, l\rangle$ does not belong to the augmented language $L^*_{\rho,\alpha(k)}$, and the claim follows by soundness of ver of Q. The probability of success of an adversary in the soundness experiment is at most $\text{ADV}(Q)$.

Composable Partial-ZK: For the first part of the definition, the language parameters are identically distributed in the two worlds since in the real-world ρ is sampled according to $\mathcal{D}$, and in the partial simulation world sfK_1 generates ρ using algorithm E_1 which samples from $\mathcal{D}$ as well. Now, the CRS in the real-world is generated by first sampling k from K_λ, and then invoking crsgen of Q on $\langle \rho, \alpha(k)\rangle$. In the partial-simulation world $\mathsf{crs\text{-}sim}$ of Q is called on the same (i.e. k is sampled from K_λ as well). Thus, indistinguishability follows from composable zero-knowledge property of Q.

For the second part of the definition, for sake of exposition, we will continue to call the first experiment as real-world, and the second as partial-simulation world (even though both have the CRS generated using sfK_1). We also define a *hybrid* experiment, where the first oracle is the simulator sfSim and the third oracle is V (as opposed to pV).

We first show that $\mathcal{A}_1$ cannot distinguish between the real-world and the hybrid-world. Since V only needs σ, which the adversary $\mathcal{A}_1$ already has access to, w.l.o.g. we can assume that the adversary does not call the third oracle. Now, even if $\mathcal{A}_1$ is given the simulation trapdoor τ, then by lemma 13, $\mathcal{A}_1$ cannot distinguish between the real-world and the hybrid-world (since with access to τ, the second oracle also becomes redundant).

Next, we show that the view of $\mathcal{A}_1$ in the hybrid-world and the partial-simulation world is statistically indistinguishable. Note that for this purpose, the first oracle is redundant for $\mathcal{A}_1$ since it can use the second oracle to get the same result. To start with, the CRS is identically generated in the two worlds. We prove the claim by induction on the number of calls to the third oracle. Let N be the total number of calls to the third oracle. Suppose, up to the $(i-1)$-th such call, the view of $\mathcal{A}_1$ is statistically indistinguishable in the two worlds. Now, consider the i-th call. Let $(x, l, (T, W))$ be the input generated by $\mathcal{A}_1$ in the hybrid-world for the i-call. By induction, w.h.p. the input generated is same in the partial-simulation world. Now, pV differs from V in that it has three conjuncts (a), (b) and (c), where (c) is same as V. Recall, conjunct (a) is $(x \in L_\rho)$ and conjunct (b) is $(T = H_k(x, l)$. Thus, pV will differ from V only if V returns 1 and (a) or (b) does not hold. By Lemma 14, w.h.p. the output of the oracle in the hybrid-world, generated as $\mathsf{V}(\sigma, x, l, (T, W))$ is 1 only if $x \in L_\rho$ and $T = H_k(x)$, which implies (a) and (b) hold with high probability when (c) is true. Thus, the output of pV in the partial-simulation world is same as output of V in the hybrid world, w.h.p, which completes the induction step.

The difference in the probabilities in the real-world and the partial-simulation world is at most $N_3 \cdot \text{ADV}^{\text{wss}}(Q) + \text{ADV}^{\text{zk}}(Q)$.

Unbounded Partial-Simulation Soundness: This follows trivially, as pV directly verifies that s is in L_ρ using algorithm E_2 and the auxiliary information ψ in the trapdoor, and moreover it also checks that real-world verifier accepts as well.

One-Time Full-ZK: We will show that the one-time full-ZK property holds statistically. We will define a sequence of experiments and show that the view of the adversary is statistically indistinguishable in every two consecutive experiments. The first experiment $\mathbf{H}_0$ is identical to the partial-simulation world. First, note that ρ is identically generated using $\mathcal{D}$ in both worlds. Next, note that the CRS σ and trapdoors τ generated by sfK_1 is identically distributed to the CRS σ and both the trapdoors τ and τ_1 generated by otfK_1.

The next experiment $\mathbf{H}_1$ is identical to $\mathbf{H}_0$ except that on $\mathcal{A}_3$ supplied input (x^*, l^*, β^*) the proof π^* generated by sfSim is replaced by proof generated by otfSim. If β^* provided by $\mathcal{A}_3$ is not the valid membership bit for x^* then both experiments abort. So, assume that β^* is the correct membership bit. In case $\beta^* = 1$, both sfSim and otfSim behave identically. When $\beta^* = 0$, the random T^* produced by sfSim is identically distributed to the T^* generated by $H_k(x^*, l^*)$ since H is assumed to be smooth.

The next experiment $\mathbf{H}_2$ is identical to $\mathbf{H}_1$ except that the second oracle is replaced by sfV (from being pV). In order to show that the view of the adversary is indistinguishable in experiments $\mathbf{H}_2$ and $\mathbf{H}_1$, we define several hybrid experiments $\mathbf{H}_{1,i}$ (for $0 \leq i \leq N$, where N is the total number of calls to the second-oracle by $\mathcal{A}_3$ and $\mathcal{A}_4$ combined). Experiment $\mathbf{H}_{1,0}$ is identical to $\mathbf{H}_1$, and the intermediate experiments are defined inductively, by modifying the response of one additional second-oracle call starting with the last (N-th) second-oracle call, and ending with the changed response of the first second-oracle call. The last hybrid experiment $\mathbf{H}_{1,N}$ will then be same as $\mathbf{H}_2$. The second-oracle call response in experiment $\mathbf{H}_{1,i+1}$ differs only in the $(N-i)$-th second-oracle call response in $\mathbf{H}_{1,i}$. In the latter experiment, this call is still served as in $\mathbf{H}_1$ (i.e. using pV). In the former experiment $\mathbf{H}_{1,i+1}$, the $(N-i)$-th call is responded to as defined in $\mathbf{H}_2$ above (i.e. using sfV).

To show that the view of the adversary is statistically indistinguishable in $\mathbf{H}_{1,i}$ and $\mathbf{H}_{1,i+1}$, first note that the view of the adversary ($\mathcal{A}_3$ and A_4 combined) till it's $(N-i)$-th call in both experiments is identical. Moreover, as we next show, the dependence on k of this partial view (i.e. till the $(N-i)$-th call) is limited to $\alpha(k)$ and at most one evaluation of H_k (by otfSim) on an input that is not in L_ρ. To start with, the CRS generated by sfK_1 depends only on $\alpha(k)$. Next, the first oracle sfSim produces T using H_k on its input only if the membership bitβ is 1 and correct, and since H is projective this hash value is then completely determined by $\alpha(k)$. Finally, all calls to the second oracle till the $(N-i)$-th call are still served using pV, and again using the projective property of H, it is clear that the conjunct (b) in pV can be computed using only $\alpha(k)$, because for non L_ρ members, the conjunct (a) is already false, and hence (b) is redundant.

Now, the difference in the $(N-i)$-th call is that the conjunct (a) of pV is missing in sfV. Let x, l, T, W be the input supplied by the Adversary to this call. If $H_k(x,l)$ is not equal to the supplied T, then both pV and sfV return 0. So, suppose $H_k(x,l)$ is equal to T, and yet x is not in L_ρ, i.e. conjunct (a) of pV is false. First, if this input x, l, T, W is same as (x^*, l^*, T^*, W^*) associated with the one-time call to otfSim, then the experiment aborts. Thus, we can assume that this is a different input. If (x,l) is same as (x^*, l^*), then $(T,W) \neq (T^*, W^*)$. Now, by construction (i.e. by definition of otfSim) $T^* = H_k(x^*, l^*)$, and hence either $T \neq H_k(x,l)$ which is not possible by hypothesis, or $(x,l,T) = (x^*, l^*, T^*)$ and $W \neq W^*$. But, W^* is proof generated by the simulator of Q, and since the simulator of Q generates unique acceptable proofs (by assumption), the verifier ver of Q rejects (x,l,T,W), and thus both pV and sfV return 0.

On the other hand, if $(x,l) \neq (x^*, l^*)$ then by the ϵ-universal$_2$ property of H, the probability of T being same as $H_k(x,l)$ is at most ϵ. Thus, both pV and sfV return 0. That completes the induction step, and thus the view of the adversary in experiments $\mathbf{H}_1$ and $\mathbf{H}_2$ is statistically indistinguishable.

The next experiment $\mathbf{H}_3$ is identical to $\mathbf{H}_2$ except that the CRS is generated using otfK_1. The only difference is that the (verifier) trapdoor does not include ρ, ψ. But, since the second oracle is served by sfV and it does not need ρ, ψ, the experiment $\mathbf{H}_3$ is well-defined and statistically indistinguishable from $\mathbf{H}_2$, Further, $\mathbf{H}_3$ is identical to the one-time simulation world, and that completes the proof.

The statistical distance between the views of the adversaries $(\mathcal{A}_3, \mathcal{A}_4)$ in $\mathbf{H}_0$ and $\mathbf{H}_3$ is at most $(\epsilon + \delta) * (1 + M_2)$.

□ □

B.1 DSS-QA-NIZK Instantiation for Linear Subspaces

Consider a bilinear group $\mathbb{G}$ of prime order q, supporting a pairing operation $e : \mathbb{G} \times \mathbb{G}_2 \to \mathbb{G}_T$. In this section we show that linear subspace languages of the form $L_{\mathbf{A}} = \{\vec{x} \cdot \mathbf{A} \mid \vec{x} \in \mathbb{Z}_q^t\}$, where $\mathbf{A}$ is drawn from a robust witness samplable distribution $\mathcal{D}$ over $\mathbb{G}^{t \times n}$ and $L_{\mathbf{A}} \subsetneq \mathbb{G}^n$, satisfy the conditions enumerated in Section 4 and subsequently support the construction of DSS-QA-NIZKs. We first describe the construction of a ϵ-**smooth** and ϵ-**universal$_2$** (labeled) projective hash proof system for $L_{\mathbf{A}}$.

Projective Hash. We define the superset $X \supsetneq L_{\mathbf{A}}$ as $\mathbb{G}^n$, where n is the column length of $\mathbf{A}$. The key set K is $\mathbb{Z}_q^n \times \mathbb{Z}_q^n$ and the set of projection keys S is $\mathbb{G}^t \times \mathbb{G}^t$. The projection map α is given as:

$$\alpha(\vec{u}^{1\times n}, \vec{v}^{1\times n}) := (\mathbf{A} \cdot \vec{u}^\top, \mathbf{A} \cdot \vec{v}^\top)$$

Let CRHF be a collision resistant hash mapping from any domain to $\mathbb{Z}_q$. The keyed (labeled) hash function H_k is:

$$H_{(\vec{u},\vec{v})}(\vec{\mathbf{l}}^{1\times n}, l) := \vec{\mathbf{l}} \cdot \left(\vec{u}^\top + t \cdot \vec{v}^\top\right), \qquad \text{where } t = \textsc{crhf}(\vec{\mathbf{l}}, l)$$

The projected hash computation for $\vec{\mathbf{l}} = \vec{x} \cdot \mathbf{A}$ is:

$$\hat{H}_{\left(\mathbf{A}\cdot\vec{u}^\top, \mathbf{A}\cdot\vec{v}^\top\right)}(\vec{x}\cdot\mathbf{A}, l) := \vec{x}\cdot\left(\mathbf{A}\cdot\vec{u}^\top + t\cdot\mathbf{A}\cdot\vec{v}^\top\right), \qquad \text{where } t = \textsc{crhf}(\vec{\mathbf{l}}, l)$$

We now prove that this construction is ϵ-**smooth**. We have to show that the following distributions are indistinguishable for any $\vec{\mathbf{l}} \notin L_{\mathbf{A}}$:

$$\left(\mathbf{A}\cdot\vec{u}^\top,\ \mathbf{A}\cdot\vec{v}^\top,\ \vec{\mathbf{l}}\cdot\left(\vec{u}^\top + t\cdot\vec{v}^\top\right)\right) \approx \left(\mathbf{A}\cdot\vec{u}^\top,\ \mathbf{A}\cdot\vec{v}^\top,\ \mathbf{r}\right),$$

where $\vec{u}, \vec{v}$ are uniform from $\mathbb{Z}_q^n$ and $\mathbf{r}$ is uniform from $\mathbb{G}$, independent of $\vec{u}$ and $\vec{v}$.

Let the matrix $\mathsf{N}^{n\times(n-t)}$ be a complete basis for the null-space of A. Since $L_{\mathbf{A}}$ is a proper subspace of $\mathbb{G}^n$, N is guaranteed to be non-zero. Then the left distribution is the same as, given random $\vec{r}$ from $\mathbb{Z}_q^{n-t}$:

$$\begin{aligned}&\left(\mathbf{A}\cdot(\vec{u}^\top + \mathsf{N}\cdot\vec{r}^\top),\ \mathbf{A}\cdot\vec{v}^\top,\ \vec{\mathbf{l}}\cdot\left(\vec{u}^\top + \mathsf{N}\cdot\vec{r}^\top + t\cdot\vec{v}^\top\right)\right)\\ =&\left(\mathbf{A}\cdot\vec{u}^\top,\ \mathbf{A}\cdot\vec{v}^\top,\ \vec{\mathbf{l}}\cdot(\vec{u}^\top + t\cdot\vec{v}^\top) + \vec{\mathbf{l}}\cdot\mathsf{N}\cdot\vec{r}^\top\right)\end{aligned}$$

Since $\vec{\mathbf{l}} \notin L_{\mathbf{A}}$, we have that $\vec{\mathbf{l}}\cdot\mathsf{N}$ is a non-zero vector, and hence $\vec{\mathbf{l}}\cdot\mathsf{N}\cdot\vec{r}^\top$ is uniformly distributed in $\mathbb{G}$, independent of $\vec{u}$ and $\vec{v}$. Thus ϵ-**smooth**-ness follows.

We now prove that this construction is ϵ-**universal$_2$**. To do that it is sufficient to prove that the following distributions are indistinguishable for any $\vec{\mathbf{l}} \in X$ and $\vec{\mathbf{l}}^* \in X\backslash L_{\mathbf{A}}$, with $\vec{\mathbf{l}}^* \neq \vec{\mathbf{l}}$:

$$\begin{aligned}&\left(\mathbf{A}\cdot\vec{u}^\top,\ \mathbf{A}\cdot\vec{v}^\top,\ \vec{\mathbf{l}}\cdot\left(\vec{u}^\top + t\cdot\vec{v}^\top\right),\ \vec{\mathbf{l}}^*\cdot\left(\vec{u}^\top + t^*\cdot\vec{v}^\top\right)\right) \approx\\ &\left(\mathbf{A}\cdot\vec{u}^\top,\ \mathbf{A}\cdot\vec{v}^\top,\ \vec{\mathbf{l}}\cdot\left(\vec{u}^\top + t\cdot\vec{v}^\top\right),\ \mathbf{r}\right),\end{aligned}$$

where $\vec{u}, \vec{v}$ are uniform from $\mathbb{Z}_q^n$ and $\mathbf{r}$ is uniform from $\mathbb{G}$.

The left distribution is the same as, given random $\vec{r}$ from $\mathbb{Z}_q^{n-t}$:

$$\begin{aligned}&\left(\begin{array}{c}\mathbf{A}\cdot(\vec{u}^\top + \mathsf{N}\cdot t\cdot\vec{r}^\top), \mathbf{A}\cdot(\vec{v}^\top - \mathsf{N}\cdot\vec{r}^\top),\\ \vec{\mathbf{l}}\cdot\left(\vec{u}^\top + t\cdot\vec{v}^\top + \mathsf{N}\cdot(t\cdot\vec{r}^\top - t\cdot\vec{r}^\top)\right), \vec{\mathbf{l}}^*\cdot\left(\vec{u}^\top + t^*\cdot\vec{v}^\top + \mathsf{N}\cdot(t\cdot\vec{r}^\top - t^*\cdot\vec{r}^\top)\right)\end{array}\right)\\ =&\left(\mathbf{A}\cdot\vec{u}^\top,\ \mathbf{A}\cdot\vec{v}^\top,\ \vec{\mathbf{l}}\cdot\left(\vec{u}^\top + t\cdot\vec{v}^\top\right),\ \vec{\mathbf{l}}^*\cdot\left(\vec{u}^\top + t^*\cdot\vec{v}^\top\right) + (t - t^*)\cdot\vec{\mathbf{l}}^*\cdot\mathsf{N}\cdot\vec{r}^\top\right)\end{aligned}$$

Since $\vec{\mathbf{l}}^* \notin L_{\mathbf{A}}$, we have that $\vec{\mathbf{l}}^*\cdot\mathsf{N}$ is a non-zero vector and also whp $t^* \neq t$ (since $\vec{\mathbf{l}}^* \neq \vec{\mathbf{l}}$), and hence $(t-t^*)\cdot\vec{\mathbf{l}}^*\cdot\mathsf{N}\cdot\vec{r}^\top$ is uniformly distributed in $\mathbb{G}$, independent of $\vec{u}$ and $\vec{v}$. Thus ϵ-**universal$_2$**-ness follows.

Given the above Projective Hash construction, we note that the augmented language corresponding to $L_{\mathbf{A}}$ is $L^*_{\mathbf{A},\vec{\mathbf{s}}_1,\vec{\mathbf{s}}_2}$ defined as follows:

$$L^*_{\mathbf{A},\vec{\mathbf{s}}_1,\vec{\mathbf{s}}_2} = \{(\vec{\mathbf{l}}, T, l) \mid \exists \vec{x} \in \mathbb{Z}_q^t : \vec{\mathbf{l}} = \vec{x}\cdot\mathbf{A}, T = \vec{x}\cdot(\vec{\mathbf{s}}_1^\top + t\cdot\vec{\mathbf{s}}_2^\top), t = \textsc{crhf}(\vec{\mathbf{l}}, l)\}$$

This is a tagged linear subspace language admitting a QA-NIZK proof consisting of k group elements [JR14] under the k-linear assumption. To fulfill the requirements of Section 4, we show that in general the QA-NIZK of [JR14] is composable zero-knowledge and weakly simulation-sound. We also note that the construction generates unique proofs as the prover is deterministic. Further, it is easy to see that all proofs generated by simulator (even on fake tuples) are accepted by the verifier. Since, the verification test is linear, it follows that **simulator generates unique acceptable proofs**.

Composable ZK. The Real World CRS and the Simulation World CRS are information theoretically indistinguishable by construction. Also the Real World prover and Simulator are information theoretically identical as functions on distributions, when the input is a language member. So no adversary, even given the simulation trapdoor, can distinguish between the oracles. Hence the QA-NIZK construction is composable zero-knowledge.

Weakly Simulation-Sound. We give a proof intuition for this by hopping through a sequence of games. Game G_0 follows the statement of the requirement, that is, $\mathcal{A}$ has access to the proof simulator, provided it calls with only valid language members, and wins if its output (x, l, π) passes verification and $x \notin L_\rho$. Let the winning probability be denoted Δ. In game G_1, the language parameter ρ is generated from the witness samplable distribution $\mathcal{D}$ and its witness ψ is retained by the challenger. The challenger then computes the simulation CRS along with the trapdoor and instantiates the oracle proof simulator with them. Next, in game G_2, the CRS is generated in a statistically identical manner using the null space of the language parameter witness ψ, as in the proof of soundness of the QA-NIZK in [JR14]. We note that G_2 is statistically indistinguishable from G_0 via G_1. Hence $\mathcal{A}$'s probability to win in G_2 is $\Delta + \epsilon$, where ϵ is negligible in the security parameter.

Consider an event where $\mathcal{A}$ wins in G_2. In [JR14], it was shown that, with the challenger possessing the language parameter witness ψ, this can enable construction of a k-linear adversary with the same advantage as $\mathcal{A}$. Therefore $\Delta + \epsilon \leq \text{ADV}_{\text{k-linear}}$. Therefore, Δ itself is negligible in the security parameter.

Sampling Algorithms. We make the assumption that algorithms E_1, E_2 exist such that the distribution $\mathcal{D}$ is efficiently witness samplable using (E_1, E_2). A simple example of such a $\mathcal{D}$ is the uniform distribution on $\mathbb{G}^{t\times n}$. Then E_1 is the algorithm which samples A uniformly from $\mathbb{Z}_q^{t\times n}$ and sets $\mathbf{A} = \mathsf{A}\cdot \mathbf{g}$. E_2 is the algorithm which given language candidate $\vec{\mathbf{l}}$ and A decides whether $\vec{\mathbf{l}}$ is in the language as follows: It computes N which is a complete basis for the nullspace of A and outputs YES if $\vec{\mathbf{l}} \cdot \mathsf{N} \stackrel{?}{=} 0$ and NO otherwise.

Finally, E_3 is simply a sampler for the uniform distribution on $\Pi = \mathbb{G}$.

B.2 DSS-QA-NIZK for Specific Languages

In this section we give examples of our DSS-QA-NIZK generic construction for popular linear subspace languages which are used widely in the literature. We will consider bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, with an efficiently computable pairing e from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$. Each group will be assumed to be cyclic with prime order q. We also switch to *multiplicative* notation for better readability.

We consider Diffie-Hellman (DH) languages in group $\mathbb{G}_1$, assuming the DDH assumption holds in $\mathbb{G}_2$. We first recap the assumption.

Definition 15 (DDH [DH76]) *Assuming a generation algorithm $\mathcal{G}$ that outputs a tuple $(q, \mathbb{G}, \mathbf{g})$ such that $\mathbb{G}$ is of prime order q and has generator g, the DDH assumption asserts that it is computationally infeasible to distinguish between $(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^b, \mathbf{g}^c)$ and $(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^b, \mathbf{g}^{ab})$ for $a, b, c \leftarrow \mathbb{Z}_q$. More formally, for all PPT adversaries A there exists a negligible function $\nu()$ such that*

$$\left| \begin{array}{c} Pr[(q, \mathbb{G}, \mathbf{g}) \leftarrow \mathcal{G}(1^m); a, b, c \leftarrow \mathbb{Z}_q : A(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^b, \mathbf{g}^c) = 1] - \\ Pr[(q, \mathbb{G}, \mathbf{g}) \leftarrow \mathcal{G}(1^m); a, b \leftarrow \mathbb{Z}_q : A(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^b, \mathbf{g}^{ab}) = 1] \end{array} \right| < \nu(m)$$

B.2.1 DSS-QA-NIZK for DH Languages.

We consider the following class of languages parametrized by $\mathbf{g}, \mathbf{a}$ ($\in \mathbb{G}_1$). The **DH language** corresponding to one such parameter $\rho = \langle \mathbf{g}, \mathbf{a} \rangle$ is $L_\rho = \{\langle \mathbf{g}^x, \mathbf{a}^x \rangle \mid x \in \mathbb{Z}_q\}$. The distribution $\mathcal{D}$ under which the quasi-adaptive NIZK is obtained is defined by picking $\mathbf{g}_1$ as a generator for $\mathbb{G}_1$ according to a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ generation algorithm for which the DDH assumption holds for $\mathbb{G}_2$, and then choosing a randomly from $\mathbb{Z}_q$ and letting $\mathbf{g} = \mathbf{g}_1, \mathbf{a} = \mathbf{g}_1^a$. The components are:

Projective Hash: We define the superset $X \supsetneq L_\rho$ as $\mathbb{G}_1^2$. The key set K is $\mathbb{Z}_q^4$ and the set of projection keys S is $\mathbb{G}_1^2$. The projection map α is given as:

$$\alpha(d_1, d_2, e_1, e_2) := (\mathbf{g}^{d_1}\mathbf{a}^{d_2}, \mathbf{g}^{e_1}\mathbf{a}^{e_2})$$

Let CRHF be a collision resistant hash mapping from any domain to $\mathbb{Z}_q$. The keyed (labeled) hash function H_k is:

$$H_{(d_1,d_2,e_1,e_2)}(\mathbf{x}_1, \mathbf{x}_2, l) := \mathbf{x}_1^{d_1+te_1}\mathbf{x}_2^{d_2+te_2}, \qquad \text{where } t = \textsc{crhf}(\mathbf{x}_1, \mathbf{x}_2, l)$$

The public hash computation for $\vec{\mathbf{l}} = (\mathbf{g}^x, \mathbf{a}^x)$ is:

$$\hat{H}_{(\mathbf{d},\mathbf{e})}(\mathbf{g}^x, \mathbf{a}^x, l) := (\mathbf{d}\mathbf{e}^t)^x, \qquad \text{where } t = \textsc{crhf}(\mathbf{g}^x, \mathbf{a}^x, l)$$

QA-NIZK: Given the above Projective Hash construction, we note that the augmented language corresponding to L_ρ is $L^*_{\rho,\mathbf{d},\mathbf{e}}$ defined as follows:

$$L^*_{\rho,\mathbf{d},\mathbf{e}} = \{(\mathbf{x}_1, \mathbf{x}_2, T, l) \mid \exists x \in \mathbb{Z}_q : \mathbf{x}_1 = \mathbf{g}^x, \mathbf{x}_2 = \mathbf{a}^x, T = (\mathbf{d}\mathbf{e}^t)^x, t = \textsc{crhf}(\mathbf{x}_1, \mathbf{x}_2, l)\}$$

This is a tagged linear subspace language admitting a QA-NIZK proof consisting of 1 group elements [JR14] under the *DDH* assumption. The augmented parameter distribution $\mathcal{D}'$ is the uniform distribution on $\mathbb{G}_1^4$, with the witnesses being the discrete logs w.r.t. $\mathbf{g}_1$.

- The algorithm K_0 is just the group generation algorithm (it takes a unary string 1^m as input), and it also generates a collision-resistant hash function CRHF. The CRS generation algorithm K_1 takes language parameter $\langle \mathbf{g}, \mathbf{a}, \mathbf{d}, \mathbf{e} \rangle$ and other bilinear group parameters as input and generates the CRS as follows: it picks c, b, u_1, u_2 randomly and independently from $\mathbb{Z}_q$ and sets the CRS to be $(\mathsf{CRS}_p, \mathsf{CRS}_v, \textsc{crhf})$:

$$\mathsf{CRS}_p = \left\{ \ \mathbf{w}_1 = \mathbf{g}^{u_1}\mathbf{a}^{cb^{-1}}\mathbf{d}^{b^{-1}}, \ \mathbf{w}_2 = \mathbf{g}^{u_2}\mathbf{e}^{b^{-1}} \ \right\}$$
$$\mathsf{CRS}_v = \left\{ \ \mathbf{g}_2, \ \mathbf{b} = \mathbf{g}_2^b, \ \mathbf{c} = \mathbf{g}_2^c, \ \mathbf{v}_1 = \mathbf{g}_2^{bu_1}, \ \mathbf{v}_2 = \mathbf{g}_2^{bu_2} \ \right\}.$$

- The **prover** P takes as input CRS_p, a language member $\langle \mathbf{x}_1, \mathbf{x}_2, T, l \rangle$ and its witness x and produces a proof π consisting of one $\mathbb{G}_2$ element W as follows: Compute $\iota = \textsc{crhf}(\mathbf{x}_1, \mathbf{x}_2, l)$. Next, compute $W = (\mathbf{w}_1\mathbf{w}_2^\iota)^x$.
- The **verifier** V takes as input CRS_v, a potential language member $\langle \mathbf{x}_1, \mathbf{x}_2, T, l \rangle$, and a proof $\pi = W$, computes $\iota = \textsc{crhf}(\mathbf{x}_1, \mathbf{x}_2, l)$, and outputs true iff

$$e(\mathbf{x}_1, \mathbf{v}_1\mathbf{v}_2^\iota) \cdot e(\mathbf{x}_2, \mathbf{c}) \cdot e(T, \mathbf{g}_2) \stackrel{?}{=} e(W, \mathbf{b})$$

B.2.2 Summary - DSS-QA-NIZK.

Thus, summing up, the DSS-QA-NIZK system is as follows - the **real world** components are as follows:

CRS generators: The algorithm K_0 is just the group generation algorithm (it takes a unary string 1^m as input and outputs λ), and it also generates a collision-resistant hash function $\textsc{crhf}$. Then $\rho = (\mathbf{g}, \mathbf{a})$ is sampled from $\mathcal{D}_\lambda$. The CRS generation algorithm K_1 takes language parameter ρ and other bilinear group parameters as input and generates the CRS as follows: it picks d, e, c, b, u_1, u_2 randomly and independently from $\mathbb{Z}_q$ and sets the CRS to be $(\mathsf{CRS}_p, \mathsf{CRS}_v, \textsc{crhf})$:

$$\mathsf{CRS}_p = \left\{ \ \mathbf{d} = \mathbf{g}^d, \mathbf{e} = \mathbf{g}^e, \mathbf{w}_1 = \mathbf{g}^{u_1}\mathbf{a}^{cb^{-1}}\mathbf{d}^{b^{-1}}, \ \mathbf{w}_2 = \mathbf{g}^{u_2}\mathbf{e}^{b^{-1}} \ \right\}$$
$$\mathsf{CRS}_v = \{ \ \mathbf{g}_2, \ \mathbf{b} = \mathbf{g}_2^b, \ \mathbf{c} = \mathbf{g}_2^c, \ \mathbf{v}_1 = \mathbf{g}_2^{bu_1}, \ \mathbf{v}_2 = \mathbf{g}_2^{bu_2} \ \}.$$

Prover: The prover P takes as input CRS_p, a language member $(\mathbf{x}_1, \mathbf{x}_2)$ and its witness x and a label l and produces a proof π consisting of two $\mathbb{G}_2$ elements T, W as follows: Compute $\iota = \textsc{crhf}(\mathbf{x}_1, \mathbf{x}_2, l)$. Next, compute $T = (\mathbf{d}\mathbf{e}^\iota)^x$ and $W = (\mathbf{w}_1\mathbf{w}_2^\iota)^x$. Output $\pi = (T, W)$.

Verifier: The verifier V takes as input CRS_v, a potential language member $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle$, and a proof $\pi = (T, W)$, computes $\iota = \textsc{crhf}(\mathbf{x}_1, \mathbf{x}_2)$, and outputs true iff

$$e(\mathbf{x}_1, \mathbf{v}_1\mathbf{v}_2^\iota) \cdot e(\mathbf{x}_2, \mathbf{c}) \cdot e(T, \mathbf{g}_2) \stackrel{?}{=} e(W, \mathbf{b})$$

The **partial simulation world** components are as follows:

Semi-functional CRS simulator: The CRS generation algorithm sfK_1 takes the bilinear group parameters as input and samples $a \leftarrow \mathbb{Z}_q$ and sets $\rho = (\mathbf{g}, \mathbf{a}) = (\mathbf{g}_1, \mathbf{g}_1^a)$. It then generates the CRS as follows: it picks $d_1, d_2, e_1, e_2, c, b, u_1, u_2$ randomly and independently from $\mathbb{Z}_q$ and sets the CRS to be $(\mathsf{CRS}_p, \mathsf{CRS}_v, \textsc{crhf})$:

$$\mathsf{CRS}_p = \left\{ \ \mathbf{d} = \mathbf{g}^{d_1}\mathbf{a}^{d_2}, \mathbf{e} = \mathbf{g}^{e_1}\mathbf{a}^{e_2}, \mathbf{w}_1 = \mathbf{g}^{u_1}\mathbf{a}^{cb^{-1}}\mathbf{d}^{b^{-1}}, \ \mathbf{w}_2 = \mathbf{g}^{u_2}\mathbf{e}^{b^{-1}} \ \right\}$$
$$\mathsf{CRS}_v = \{ \ \mathbf{g}_2, \ \mathbf{b} = \mathbf{g}_2^b, \ \mathbf{c} = \mathbf{g}_2^c, \ \mathbf{v}_1 = \mathbf{g}_2^{bu_1}, \ \mathbf{v}_2 = \mathbf{g}_2^{bu_2} \ \}.$$

Set the trapdoors as follows: $\tau = (b, c, d_1, d_2, e_1, e_2, u_1, u_2)$, $\eta = (a, d_1, d_2, e_1, e_2)$.

Semi-functional Simulator: sfSim uses trapdoor τ to produce a (partially-simulated) proof for a potential language member $(\mathbf{x}_1, \mathbf{x}_2)$, a label l and a binary bit β as follows: Compute $\iota = \textsc{crhf}(\mathbf{x}_1, \mathbf{x}_2, l)$.

If $\beta = 1$, output:

$$T = \mathbf{x}_1^{d_1+\iota e_1}\mathbf{x}_2^{d_2+\iota e_2}, \ W = \mathbf{x}_1^{u_1+\iota u_2} \ \mathbf{x}_2^{cb^{-1}} T^{b^{-1}},$$

else sample $\mathbf{r}$ uniformly at random from $\mathbb{G}_1$ and output:

$$T = \mathbf{r} \ , \ W = \mathbf{x}_1^{u_1+\iota u_2} \ \mathbf{x}_2^{cb^{-1}} T^{b^{-1}}$$

Private Verifier: pV uses trapdoor η to check a potential language member $(\mathbf{x}_1, \mathbf{x}_2)$, label l and a proof (T, W) as follows: it outputs 1 iff (a) $\mathbf{x}_1^a \stackrel{?}{=} \mathbf{x}_2$, and (b) $T \stackrel{?}{=} \mathbf{x}_1^{d_1+\iota e_1}\mathbf{x}_2^{d_2+\iota e_2}$, with $\iota = \textsc{crhf}(\mathbf{x}_1, \mathbf{x}_2, l)$, and (c) $e(\mathbf{x}_1, \mathbf{v}_1\mathbf{v}_2^\iota) \cdot e(\mathbf{x}_2, \mathbf{c}) \cdot e(T, \mathbf{g}_2) \stackrel{?}{=} e(W, \mathbf{b})$.

The **one-time full simulation** components are as follows:

One-time full-simulation CRS generator: The CRS generation algorithm otfK_1 takes the bilinear group parameters as input and samples $a \leftarrow \mathbb{Z}_q$ and sets $\rho = (\mathbf{g}, \mathbf{a}) = (\mathbf{g}_1, \mathbf{g}_1^a)$. It then generates the CRS as follows: it picks $d_1, d_2, e_1, e_2, c, b, u_1, u_2$ randomly and independently from $\mathbb{Z}_q$ and sets the CRS to be $(\mathsf{CRS}_p, \mathsf{CRS}_v, \text{CRHF})$:

$$\mathsf{CRS}_p = \left\{ \; \mathbf{d} = \mathbf{g}^{d_1}\mathbf{a}^{d_2}, \mathbf{e} = \mathbf{g}^{e_1}\mathbf{a}^{e_2}, \mathbf{w}_1 = \mathbf{g}^{u_1}\mathbf{a}^{cb^{-1}}\mathbf{d}^{b^{-1}}, \; \mathbf{w}_2 = \mathbf{g}^{u_2}\mathbf{e}^{b^{-1}} \; \right\}$$

$$\mathsf{CRS}_v = \left\{ \; \mathbf{g}_2, \; \mathbf{b} = \mathbf{g}_2^b, \; \mathbf{c} = \mathbf{g}_2^c, \; \mathbf{v}_1 = \mathbf{g}_2^{bu_1}, \; \mathbf{v}_2 = \mathbf{g}_2^{bu_2} \; \right\}.$$

Set the trapdoors as follows: $\tau = \tau_1 = (b, c, d_1, d_2, e_1, e_2, u_1, u_2)$, $\eta = (d_1, d_2, e_1, e_2)$.

One-time full simulator: otfSim uses trapdoor τ_1 to produce a (partially-simulated) proof for a potential language member $(\mathbf{x}_1, \mathbf{x}_2)$ and a label l as follows: Compute $\iota = \text{CRHF}(\mathbf{x}_1, \mathbf{x}_2, l)$. Output:

$$T = \mathbf{x}_1^{d_1+\iota e_1}\mathbf{x}_2^{d_2+\iota e_2}, \; W = \mathbf{x}_1^{u_1+\iota u_2} \; \mathbf{x}_2^{cb^{-1}} T^{b^{-1}},$$

Semi-functional verifier: sfV uses trapdoor η to check a potential language member $(\mathbf{x}_1, \mathbf{x}_2)$, label l and a proof (T, W) as follows: it outputs 1 iff (a) $T \overset{?}{=} \mathbf{x}_1^{d_1+\iota e_1}\mathbf{x}_2^{d_2+\iota e_2}$, with $\iota = \text{CRHF}(\mathbf{x}_1, \mathbf{x}_2, l)$, and (b) $e(\mathbf{x}_1, \mathbf{v}_1\mathbf{v}_2^{\iota}) \cdot e(\mathbf{x}_2, \mathbf{c}) \cdot e(T, \mathbf{g}_2) \overset{?}{=} e(W, \mathbf{b})$.

Denoting this DSS-QA-NIZK as Σ, we instantiate the concrete parameters for it's correctness and security following Appendix B:

- $\text{ADV}^{\text{comp}}(\Sigma) = 0$,
- $\text{ADV}^{\text{sound}}(\Sigma) \leq 2 \cdot \text{ADV}_{DDH} + 1/q$,
- $\text{ADV}^{\text{pzk}}(\Sigma, N_3) \leq N_3 \cdot 2 \cdot \text{ADV}_{DDH} + 1/q$,
- $\text{ADV}^{\text{pss}}(\Sigma) = 0$,
- $\text{DIST}^{\text{otzk}}(\Sigma) = 0$.

C Keyed-Homomorphic CCA Encryption

Definition 16 (KH-PKE Scheme) *A KH-PKE scheme for a message space $\mathcal{M}$ with a binary operation $\odot$ defined from $\mathcal{M}^2 \rightarrow \mathcal{M}$ is a tuple of algorithms $(KeyGen, Enc, Dec, Eval)$:*

KeyGen: *This algorithm takes 1^λ as input, and returns a public key pk, a decryption key sk_d, and a homomorphic operation key sk_h.*

Enc: *This algorithm takes pk and a message $M \in \mathcal{M}$ as inputs and returns a ciphertext C.*

Dec: *This algorithm takes sk_d and a ciphertext C as input, and returns M or $\perp$.*

Eval: *This algorithm takes sk_h and two ciphertexts C_1 and C_2 as inputs, and returns a ciphertext C or $\perp$.*

The scheme is said to be correct if (i) for Enc we have $Dec(sk_d, C) = M$ and (ii) for Eval we have $Dec(sk_d, C) = Dec(sk_d, C_1) \odot Dec(sk_d, C_2)$, where if any operand of $\odot$ is $\perp$ then the result is $\perp$.

Definition 17 (KH-CCA Security) *A KH-PKE scheme $(KeyGen, Enc, Dec, Eval)$ for a message space $\mathcal{M}$ with a binary operation $\odot$ defined from $\mathcal{M}^2 \to \mathcal{M}$ is said to be KH-CCA secure if no PPT adversary $\mathcal{A}$ has non-negligible advantage in winning the following game with a challenger:*

KeyGen: *The challenger takes 1^λ as input, and computes a public key pk, a decryption key sk_d, and a homomorphic operation key sk_h. It then gives pk to $\mathcal{A}$. It also initializes a list $D := \emptyset$. It also gives oracle access to $\mathcal{A}$ to the functions $Enc(.), Eval(sk_h, .)$, RevHK and $Dec(sk_d, .)$ defined as follows:*

Enc: *This query is performed once. On $\mathcal{A}$'s inputs $m_0, m_1 \in \mathcal{M}$, the challenger randomly samples a bit b and computes $C := Enc(pk, m_b)$ and returns C. Further, it sets $D := D \cup \{C\}$.*

Eval: *This oracle is not available after $\mathcal{A}$ has requested RevHK. On receiving ciphertexts C_1, C_2 from $\mathcal{A}$, computes $C := Eval(sk_h, C_1, C_2)$ and returns C. In addition if $C_1 \in D$ or $C_2 \in D$, then sets $D := D \cup \{C\}$.*

RevHK: *Upon receiving this request, returns sk_h.*

Dec: *This oracle is not available after $\mathcal{A}$ has both requested RevHK and obtained the challenge ciphertext C^* in any order. When available, it returns $\perp$ if $C \in D$ and $Dec(sk_d, C)$ on $\mathcal{A}$'s query C.*

The adversary $\mathcal{A}$ outputs a bit b' and wins if $b' = b$. Its advantage is defined to be $|\Pr[\mathcal{A}\ wins] - 1/2|$.

C.1 Proof of Theorem for KH-CCA Construction

We first prove that the construction in section 5 is secure assuming the Adversary does not invoke RevHK. In appendix C.2 we extend the result to include RevHK.

Let M be the number of decryption queries, and let L be the nummber of Eval queries. Let $N \leq L$ be the number of dependent Eval queries. We will use the notation from appendix B for the various probabilities for DSS-QA-NIZK Σ. In the following we will assme that the Diffie-Hellman language is parameterized by $(\mathbf{g}, \mathbf{g}^a)$, where $\mathbf{g}$ is chosen randomly from a DDH hard group of prime order q. The maximum advantage of any adversary in the DDH security game over this group will be denoted by $\textsc{adv}_{\text{DDH}}$.

Theorem 8 [Security of KH-Enc Construction] The above algorithms $\mathcal{P} = (KeyGen, Enc, Dec, Eval)$ (from section 5) constitute a KH-CCA secure Keyed-Homomorphic Public Key Encryption scheme with multiplicative homomorphism, if Σ is a DSS-QA-NIZK for the parameterized Diffie-Hellman language (with language parameters distributed randomly) and RevHK is not available. The Adversary's advantage in the KH-CCA security game is at most

$$(2 * L) * \textsc{adv}^{\text{pzk}}(\Sigma, 1) + \textsc{adv}^{\text{pzk}}(\Sigma, 2 * L) +$$
$$O(L * (M + 2 * L)) * (\textsc{dist}^{\text{otzk}}(\Sigma) + \textsc{adv}^{\text{pss}}(\Sigma)) + (N + 1) * \textsc{adv}_{\text{DDH}} + O(L/q).$$

Instantiating the DSS-QA-NIZK of the DH language as in Appendix B.2, we obtain that the Adversary' advantage above is at most $(4 * L) * (1/q + 2\text{ADV}_{DDH}) + (N+1) * \text{ADV}_{\text{DDH}} + O(L/q)$.
Proof: We will prove the theorem by going through a sequence of games, where the first game $\mathbf{G}_0$ is identical to the game in the security definition, and the challenger uses the various KH-PKE components of $\mathcal{P}$ above. The last game is a world where the challenger encrypts μ instead of m_b, for any constant μ, and hence the view of the adversary is independent of b. Note, in $\mathbf{G}_0$, the decryption returns $\perp$ if $c \in D$. We will call a call to Eval *dependent* if either c_1 or c_2 is in D. Otherwise, it will be called *independent*. Note, ciphertexts generated by dependent Eval calls get included in D.

$\boxed{\textbf{Game } \mathbf{G}_1}$: Same as game $\mathbf{G}_0$, but now during decryption the challenger switches to the private verifier pV. In all independent Eval calls, the verifier is switched to pV. Note, all dependent Eval calls continues to use real-world verifier V. Also the challenge encryption is performed using sfSim instead of P, with β set to 1 and with the witness w dropped.

Indistinguishability follows due to the partial-ZK property, but we need to prove that all calls to sfSim by Eval are with language members. This is proven by induction over the order of invocations of Eval, using the weak simulation-soundness lemma 4, and by using the multiplicative homomorphism property of DH language. Thus, the probability of the adversary distinguishing $\mathbf{G}_0$ from $\mathbf{G}_1$ is at most $2 * L * (\text{ADV}^{\text{pzk}}(\Sigma, 1) + \text{ADV}^{\text{pss}}(\Sigma)) + \text{ADV}^{\text{pzk}}(\Sigma, 2 * L)$.

$\boxed{\textbf{Game } \mathbf{G}_2}$: Same as game $\mathbf{G}_1$, but now all *dependent Eval* calls are changed as follows: It continues to do the verification checks on input ciphertexts using V, but instead of computing π using sfSim on the randomized product of ciphertexts, it just employs sfSim on a fresh random tuple $\langle \rho_3, \hat{\rho}_3, \gamma_3 \rangle$ (obtained by picking r at random and setting the tuple to be $\langle \mathbf{g}^r, (\mathbf{g}^a)^r, (\mathbf{g}^k)^r \rangle$) to get π, and outputs $c := (\rho_3, \hat{\rho}_3, \gamma_3, \pi)$.

Computational indistinguishability of games $\mathbf{G}_2$ and $\mathbf{G}_1$ is proven using a sequence of hybrid games, where in each hybrid game an additional dependent call to Eval, going from last call to first, is handled as in game $\mathbf{G}_2$. This is proven in Lemma 18 below. We also prove there that all calls to sfSim are with language members. Note that all calls to sfSim in dependent Eval calls in game $\mathbf{G}_2$ are made with fresh random language members.

$\boxed{\textbf{Game } \mathbf{G}_3}$: Same as game $\mathbf{G}_2$, but now KeyGen generates k_0, k_1 randomly and sets $k = k_0 + ak_1$ and gives $sk_d = (k_0, k_1)$. The challenge encryption is performed by letting $\gamma := m_b \cdot \mathbf{g}^{k_0 w} \mathbf{g}^{k_1 a w}$. Decryption is performed as $m := \gamma / (\rho^{k_0} \hat{\rho}^{k_1})$.

This step is indistinguishable from game $\mathbf{G}_2$, as unbounded partial-simulation soundness holds, and therefore $\hat{\rho} = \rho^a$. All calls to sfSim continue to be with language members.

$\boxed{\textbf{Game } \mathbf{G}_4}$: Same as game $\mathbf{G}_3$, but now KeyGen switches to the otfK_1 to generate CRS σ and trapdoors $\sigma, \tau, \tau_1, \eta$. The public key is set as $(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^{k_0}(\mathbf{g}^a)^{k_1}, \sigma)$. While dependent Eval continue to use V , the pV in decryption and independent Eval is switched to sfV which uses σ, η. The Challenge encryption is performed as $c := (\mathbf{g}^w, \mathbf{g}^{aw}, \gamma, \mathsf{otfSim}(\sigma, \tau_1, (\mathbf{g}^w, \mathbf{g}^{aw}), l))$, where $l := \gamma := m_b \cdot (\mathbf{g}^{k_0}(\mathbf{g}^a)^{k_1})^w$.

As before, all calls to sfSim in dependent Eval in both games are with language members. For independent Eval calls, this is proven by induction by noting that sfV is sound by lemma 12 (since one-time full-ZK property holds statistically). Also, the tuple corresponding to otfSim invocation is not invoked on pV (as the challenge ciphertext is in D). Thus, indistinguishability holds due to the one-time full-ZK property.

$\boxed{\textbf{Game } \mathbf{G}_5}$: Same as game $\mathbf{G}_4$, but now the challenge encryption is performed as follows: Generate $w, w' \leftarrow \mathbb{Z}_q$ and compute:
$c := (\mathbf{g}^w, \mathbf{g}^{aw'}, \gamma, \mathsf{otfSim}(\sigma, \tau_1, (\mathbf{g}^w, \mathbf{g}^{aw'}), l))$, where $l := \gamma := m_b \cdot \mathbf{g}^{k_0 w} \mathbf{g}^{k_1 aw'}$.

Indistinguishability holds by DDH assumption. We also prove that all calls to sfSim in $\mathbf{G}_5$ are with language members as before.

$\boxed{\textbf{Game } \mathbf{G}_6}$: The decryption key is changed to $sk_d = k = k_0 + ak_1$, and the decryption is performed as $m := \gamma / \rho^{sk_d}$.

This step is indistinguishable from game $\mathbf{G}_5$, since by soundness of sfV using lemma 12, $\hat{\rho} = \rho^a$.

$\boxed{\textbf{Game } \mathbf{G}_7}$: Public key is set as $(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^{k_0 + ak_1}, \sigma)$, with decryption key $sk_d = k = k_0 + ak_1$. Decryption is performed as $m := \gamma / \rho^{k_0 + ak_1}$ Now the challenge encryption is performed as $c := (\mathbf{g}^w, \mathbf{g}^{aw'}, \gamma, \mathsf{otfSim}(\sigma, \tau, (\mathbf{g}^w, \mathbf{g}^{aw'}), l))$, where $l := \gamma := \mu \cdot \mathbf{g}^{k_0 w} \mathbf{g}^{k_1 aw'}$, and μ is an arbitrary constant.

With a, k_0, k_1, w, w' chosen randomly and independently, it can be shown that the joint distribution of $(a, k_0 + ak_1, w, w', \log m_b + k_0 w + k_1 aw')$ is statistically indistinguishable from $(a, k_0 + ak_1, w, w', \log \mu + k_0 w + k_1 aw')$, as is standard in Cramer-Shoup CCA2-encryption [CS02].

Now, noting that the view of the adversary in game $\mathbf{G}_7$ is completely independent of b, the probability that it can predict b in $\mathbf{G}_7$ is at most half, and that completes the proof. □ □

Lemma 18 *Assuming all calls to* sfSim *in game* $\mathbf{G}_1$ *are with language members, then the view of the adversary in games* $\mathbf{G}_1$ *and* $\mathbf{G}_2$ *is computationally indistinguishable, and all calls to* sfSim *in game* $\mathbf{G}_2$ *are also with language members with probability at least* $1 - O(N*L)*(\textsc{dist}^{otzk}(\Sigma) + \textsc{adv}^{pss}(\Sigma))$. *The Adversary's probability of distinguishing the two games is at most*

$$O(N * (M + 2 * L)) * (\textsc{dist}^{otzk}(\Sigma) + \textsc{adv}^{pss}(\Sigma)) + N * \textsc{adv}_{\mathrm{DDH}} + O(N/q).$$

Proof: Let N be the maximum number of dependent calls to Eval. Then hybrid games, called game $\mathbf{G}_{1,j}$ (for $j = [0..N]$), are defined inductively as follows. Game $\mathbf{G}_{1,0}$ is same as game $\mathbf{G}_1$. For $j \in [0..N-1]$, game $\mathbf{G}_{1,j+1}$ differs from game $\mathbf{G}_{1,j}$, in that the $(N-j)$-th dependent call to Eval is handled as in game $\mathbf{G}_2$ (i.e. by invoking sfSim on a random tuple). Thus, the changes in the games are done in a backward fashion, with the first change in the last dependent call to Eval. Note N can be set arbitrarily high, and if there are no more than $n(< N)$ dependent calls, then the hybrid games beyond game $\mathbf{G}_{1,n}$ are trivially and perfectly indistinguishable. Computational indistinguishability of games $\mathbf{G}_{1,j+1}$ and $\mathbf{G}_{1,j}$ is proven in Lemma 19 below. □ □

$N *$ *lemma.*

Lemma 19 *For* $j \in [0..N-1]$, *if all calls to* sfSim *in game* $\mathbf{G}_{1,j}$ *are with language members, then games* $\mathbf{G}_{1,j+1}$ *and* $\mathbf{G}_{1,j}$ *above are computationally indistinguishable, and all calls to* sfSim *in game* $\mathbf{G}_{1,j+1}$ *are also with language members with probability at least* $1 - O(L)*(\textsc{dist}^{otzk}(\Sigma) + \textsc{adv}^{pss}(\Sigma))$. *The Adversary's probability of distinguishing the two games is at most*

$$O(M + 2 * L) * (\textsc{dist}^{otzk}(\Sigma) + \textsc{adv}^{pss}(\Sigma)) + \textsc{adv}_{\mathrm{DDH}} + O(1/q).$$

Proof: First note that in game $\mathbf{G}_{1,j}$, for $k > (N-j)$, in the k-th dependent call to Eval, the semi-functional simulator sfSim is invoked on a fresh random tuple. This is important to note because, in contrast, this is not the case for earlier dependent calls to Eval, where sfSim is invoked on a product of input ciphertexts and a random tuple. By employing the one-time full-simulation property, the invocation of sfSim in the $(N-j)$-th dependent call to Eval (with say, input x, and β) can be replaced with invocation of otfSim (with only x), as long as all other calls to sfSim have correct membership bits. However, if these other calls depend on x, the correctness of their membership bits cannot be guaranteed. Hence, it is important to make future calls to sfSim in dependent Eval calls to be completely independent.

Define game $\mathbf{H}_0$ to be same as game $\mathbf{G}_{1,j}$.

$\boxed{\textbf{Game } \mathbf{H}_1}$: Same as game $\mathbf{H}_0$, but now KeyGen switches to the otfK_1 to generate CRS σ and trapdoors $\sigma, \tau, \tau_1, \eta$. The public key is set as $(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^{k_0}(\mathbf{g}^a)^{k_1}, \sigma)$. The pV in decryption and independent Eval calls is switched to sfV which uses σ, η. The proof π in the ciphertext in the $(N-j)$-the dependent call to Eval is now generated using otfSim (with CRS σ, and trapdoor τ_1) instead of sfSim. Thus, no β is provided now for this call. All other calls to Eval, continue to use sfSim with CRS and trapdoor σ, τ.

Note, the tuple corresponding to the otfSim invocation is not called to sfV as this ciphertext is in D. Also, note that the membership bits β in the calls to sfSim in game $\mathbf{H}_0$ are 1 and are correct membership bits by hypothesis. We also need to show that the membership bits in calls to sfSim (and otfSim) are correct. This is proven by induction, noting that one-time full-ZK property holds statistically, by showing that the view of the adversary is statistically indistinguishable from view in the partial-simulation world (i.e. as in game $\mathbf{H}_0$), Thus, only language members are used in calls to sfSim (and otfSim) in game $\mathbf{H}_1$ as well (and $\beta = 1$). Thus, probability that only language members are called to sfSim in $\mathbf{H}_1$ is at least $1 - 2 * L * (\textsc{dist}^{\mathrm{otzk}}(\Sigma) + \textsc{adv}^{\mathrm{pss}}(\Sigma))$.

Thus, indistinguishability of $\mathbf{H}_0$ and $\mathbf{H}_1$ holds due to the one-time full-ZK property. The statistical distance between Adversary's view in $\mathbf{H}_0$ and $\mathbf{H}_1$ is thus at most $\textsc{dist}^{\mathrm{otzk}}(\Sigma) + 2 * L * (\textsc{dist}^{\mathrm{otzk}}(\Sigma) + \textsc{adv}^{\mathrm{pss}}(\Sigma))$.

$\boxed{\textbf{Game } \mathbf{H}_2}$: Same as game $\mathbf{H}_1$, but now KeyGen generates k_0, k_1 randomly and sets $k = k_0 + ak_1$ and gives $sk_d = (k_0, k_1)$. Decryption is performed as $m := \gamma/(\rho^{k_0}\hat{\rho}^{k_1})$. Further, the random tuple in $(N-j)$-th Eval is generated as follows: Generate $r\mathbb{Z}_q$ and compute $\langle \rho_3, \hat{\rho}_3, \gamma_3 \rangle$ to be $\langle \mathbf{g}^r, \mathbf{g}^{ar}, \mathbf{g}^{k_0 r}\mathbf{g}^{k_1 ar} \rangle$.

This step is indistinguishable from game $\mathbf{H}_1$, since sfV is sound by Lemma 12, and hence $\hat{\rho} = \rho^a$ in all decryption queries. The statistical distance between the views of the adversay in $\mathbf{H}_1$ and $\mathbf{H}_2$ is at most $(M + 2 * L) * (\textsc{dist}^{\mathrm{otzk}}(\Sigma) + \textsc{adv}^{\mathrm{pss}}(\Sigma))$.

$\boxed{\textbf{Game } \mathbf{H}_3}$: Same as game $\mathbf{H}_2$, but now the random tuple in $(N-j)$-th dependent call to Eval is generated as follows: Generate $r, r' \leftarrow \mathbb{Z}_q$ and compute: $\langle \rho_3, \hat{\rho}_3, \gamma_3 \rangle$ $\langle \mathbf{g}^r, \mathbf{g}^{ar'}, \mathbf{g}^{k_0 r}\mathbf{g}^{k_1 ar'} \rangle$. Indistinguishability holds by DDH assumption. Thus, the probability of the adversary distinguishing $\mathbf{H}_2$ and $\mathbf{H}_3$ is at most $\textsc{adv}_{\mathrm{DDH}}$.

For later use, We also need to show that all calls to sfSim are with language members in $\mathbf{H}_3$. Now, this was the case in $\mathbf{H}_2$, and hence up to the $(N-j)$-th dependent Eval call, nothing has changed. Beyond that, all dependent calls to Eval use fresh random language tuples. All independent Eval calls use sfV on their input ciphertexts, and hence by employing soundness of sfV inductively, they are language members as well. The probability that all calls to sfSim are with language members is thus at least $1 - 2 * L(\textsc{dist}^{\mathrm{otzk}}(\Sigma) + \textsc{adv}^{\mathrm{pss}}(\Sigma))$.

$\boxed{\textbf{Game } \mathbf{H}_4}$: The decryption key is changed to $sk_d = k = k_0 + ak_1$, and the decryption is performed as $m := \gamma/\rho^{sk_d}$.

This step is indistinguishable from game $\mathbf{H}_3$, since sfV is sound by lemma 12, and hence $\hat{\rho} = \rho^a$. The statistical distance between the view of the Adversary in the two games is at most $(M + 2 * L) * (\text{DIST}^{\text{otzk}}(\Sigma) + \text{ADV}^{\text{pss}}(\Sigma))$.

$\boxed{\textbf{Game } \mathbf{H}_5}$: Public key is set as $(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^{k_0+ak_1}, \sigma)$, with decryption key $sk_d = k = k_0 + ak_1$. Decryption is performed as $m := \gamma/\rho^{k_0+ak_1}$. Now, the random tuple in $(N-j)$-th Eval is generated as follows: Generate $r, r', r'' \leftarrow \mathbb{Z}_q$ and compute $\langle \rho_3, \hat{\rho}_3, \gamma_3 \rangle$ to be $\langle \mathbf{g}^r, \mathbf{g}^{ar'}, \mathbf{g}^{r''} \rangle$.

With a, k_0, k_1, r, r', r'' chosen randomly and independently, it can be shown that the joint distribution of $(a, k_0+ak_1, r, r', k_0r+k_1ar')$ is statistically indistinguishable from $(a, k_0+ak_1, r, r', r'')$ (as is standard in Cramer-Shoup CCA2-encryption [CS02]), by noting that (k_0+ak_1) and (k_0r+ka_1ar') are random and independent, given a, r, r', and $a \neq= 0$ and $r \neq r'$. Thus, $\mathbf{H}_4$ and $\mathbf{H}_5$ are statistically indistinguishable with distance at most $O(1/q)$, where q is the prime order of the Diffie-Hellman groups.

$\boxed{\textbf{Game } \mathbf{H}_6}$: Now, the ciphertext in the $(N-j)$-th dependent call to Eval is computed as follows: Generate $r, r', r'' \leftarrow \mathbb{Z}_q$ and compute $\langle \rho_3, \hat{\rho}_3, \gamma_3 \rangle$ to be $\langle \mathbf{g}^r, \mathbf{g}^{ar'}, \mathbf{g}^{r''} \rangle$. Set $(\rho, \hat{\rho}, \gamma) = (\rho_3, \hat{\rho}_3, \gamma_3)$. Call otfSim on $(\rho, \hat{\rho}, \gamma)$, to get π, and output $c = (\rho, \hat{\rho}, \gamma, \pi)$.

Indistinguishability from $\mathbf{H}_5$ follows because all three components of $\langle \rho_3, \hat{\rho}_3, \gamma_3 \rangle$ are random and independent, and also independent of input ciphertexts c_1 and c_2 (of this Eval call).

We now employ a series of games similar to the above games employed backwards, and change the random tuple $\langle \rho_3, \hat{\rho}_3, \gamma_3 \rangle$ back to being a valid language tuple, and this would correspond to game $\mathbf{G}_{1,j+1}$. Moreover, the call to sfSim in the $(N-j)$-th dependent call to Eval is again with a language member. □ □

C.2 Revealing the Partial-Simulation Key

In some applications, the partial-simulation trapdoor τ can be revealed to the adversary, adaptively on adversary's demand. This event will be referred to as the *Reveal* event. In this case we would like to have the property that partial-ZK and one-time full-ZK simulation is still possible under the restriction that the adversary *do not* have access to the verifier oracle (i.e. third oracle in partial-ZK and second oracle in one-time full ZK experiments) *after* the reveal event. However, this stronger notion of security may not be easily obtainable (if at all).

For the general construction of Section 4, while the partial-ZK property with the above restriction continues to hold, the one-time full ZK property holds only with another minor restriction. The additional restriction is that the tuple (x^*, l^*, β^*) on which one-time full-simulation is required must now have $x \in L_\rho$ (and $\beta = 1$). In other words, one *cannot* go back and forth between the partial-simulation world and one-time full simulation world with fake tuples (the tuples invoked on the first oracle sfSim can continue to be fake as long as their membership-bit is correct). But, this restriction is remedied by the fact that we can show that the semi-functional verifier sfV continues to be sound. In other words, the statement of lemma 12 continues to hold for the general construction.

Thus, we define the stronger notion of DSS-QA-NIZK as follows:

Definition 20 *A dual-system non-interactive proof with (partial-simulation trapdoor) reveal oracle* *is called a* **strong** *dual-system simulation-sound quasi-adaptive NIZK (***strong DSS-QA-NIZK***) with the following changes to the DSS-QA-NIZK definition:*

The composable partial-ZK property (part one) continues to hold.

The composable partial-ZK property (part two) holds under the additional restriction that the adversary cannot call the third oracle after the reveal event.

The unbounded partial-simulation soundness continues to hold.

The trapdoors τ and τ_1 output by otfK_1 are same and statistically indistinguishable from τ output by sfK_1.

The one-time full-ZK holds under the additional restriction that (x^*, l^*, β^*) is such that $x^* \in L_\rho$ and $\beta^* = 1$, and the second oracle is not invoked after the reveal event.

The soundness of sfV as in Lemma 12 statement holds under the additional restriction that the second oracle is not invoked after the reveal event. Note, there is no restriction that (x^*, l^*, β^*) is such that $x^* \in L_\rho$ and $\beta^* = 1$.

Theorem 21 *For a parameterized class of languages $\{L_\rho\}_{\rho \in \mathsf{Lpar}}$ with probability distribution $\mathcal{D}$, if the four conditions (of Section 4) hold for projective hash family H, QA-NIZK Q, and efficient algorithms E_1, E_2, E_3, then the dual-system non-interactive proof system Σ (of Section 4) with adaptive adversarial access to reveal oracle is a strong DSS-QA-NIZK for $\{L_\rho\}_{\rho \in \mathsf{Lpar}}$ with probability distribution $\mathcal{D}$.*

Proof: As a general consideration, suppose an Adversary makes multiple outputs. Note that soundness conditions state that the probability of Adversary outputting a tuple satisfying some relation is negligible. Thus, if the Adversary has not initiated the reveal event till a particular output, then soundness continues to hold for that output.

The composable partial-ZK property (part one): by construction of sfK_1, the language parameter ρ are statistically indistinguishable in the real-world and the partial-ZK world. Further the $\alpha(k)$ part of the CRS σ is identical. Finally, the CRS generated by the CRS generators of Q are statistically indistinguishable by hypothesis.

The composable partial-ZK property (part two) under additional restriction mentioned above: Recall in the proof of theorem 6, we defined a *hybrid* experiment, where the first oracle is the simulator sfSim and the third oracle is V (as opposed to pV). Indistinguishability of the real-world and the hybrid world continues to hold as it was proven there even with adversary given access to τ.

For showing statistical indistinguishability of the hybrid-world and the partial-simulation world, we first note that (by above general consideration) the QA-NIZK Q continues to be weakly-simulation sound till the reveal event (note, revealing τ also reveals the simulation trapdoor generated by $\mathsf{crs\text{-}sim}$ of Q). Thus lemma 14 also continues to hold till the reveal event. Hence, as in proof of theorem 6, the statistical indistinguishability of the hybrid-world and the partial-simulation world holds under the restriction that the third oracle is not invoked after the reveal event.

The unbounded partial-simulation soundness holds: This follows trivially.

The trapdoors τ and τ_1 output by otfK_1 are same and statistically indistinguishable from τ output by sfK_1: all three are generated identically, given that E_1 samples from $\mathcal{D}$.

The one-time full-ZK holds under additional restrictions mentioned above: We go through the various experiments defined in proof of Theorem 6. For the indistinguishability of experiments $\mathbf{H}_1$ and $\mathbf{H}_2$, note that we have now restricted β^* to be one. Hence, if τ is revealed (in which case the private hash key k would be revealed) sfSim and otfSim still behave the same. The rest of the experiments are indistinguishable since they involve the verifier oracle (i.e. the second oracle), and these oracles are not even allowed to be called after the reveal event.

The soundness of sfV as in Lemma 12 statement holds under the additional restriction mentioned above: Since there is no restriction that (x^*, l^*, β^*) is such that $x^* \in L_\rho$ and $\beta^* = 1$, we cannot now go back to the partial-simulation world as done in the general proof of lemma 12.

Define a hybrid experiment that is identical to the experiment $\mathbf{H}_1$ in the proof of Theorem 6. In other words, it is same as the partial-simulation world, except that the proof π^* on input (x^*, l^*, β^*) is generated using otfSim (recall, the CRS and simulation trapdoors are identically distributed). Now, note that as long as the second oracle is not invoked after the reveal event, the proof that $\mathbf{H}_1$ is (almost) identically distributed to the one-time simulation world still holds (one needs to go through all the intermediate experiments as in proof of theorem 6). But note that pV is sound even in experiment $\mathbf{H}_1$. Thus, the argument in the proof of lemma 12 still holds, and hence sfV is sound except for the the tuple (x^*, l^*, π^*).

□

Note that, the weak-simulation soundness of the verifier V holds (as in Lemma 14 statement) if the Adversary has not initiated the reveal event: the proof is same as the proof of lemma 14 noting that partial-zk property (second part) holds (as shown above) till the time that the adversary outputs.

We are now ready to prove the stronger result for the KH-PKE construction of Section 5.

Theorem 22 (KH-CCA under Reveal Event) *The algorithms $\mathcal{P}$= (KeyGen, Enc, Dec, Eval) constitute a KH-CCA secure Keyed-Homomorphic Public Key Encryption scheme with multiplicative homomorphism, if Σ is a strong DSS-QA-NIZK for the parameterized Diffie-Hellman language (with language parameters distributed randomly) and Dec and Eval after not available after reveal event. The Adversary's advantage in the KH-CCA security game is at most*

$$(2 * L) * \textsc{adv}^{pzk}(\Sigma, 1) \ + \textsc{adv}^{pzk}(\Sigma, 2 * L) +$$
$$O(L * (M + 2 * L)) * (\textsc{dist}^{otzk}(\Sigma) + \textsc{adv}^{pss}(\Sigma)) \ + (N + 1) * \textsc{adv}_{\text{DDH}} + O(L/q).$$

The proof is essentially the same as the proof of Theorem 8.

Key-Reveal and CCA1 Security. We can easily extend the keyed-homomorphic scheme of Section 5 to maintain CCA1-security when the key is revealed, but the construction requires use of another group element. More specifically, a smooth projective hash (which need not be universal$_2$) of the Diffie-Hellman tuple is included, and the decryption key includes the keys to this smooth projective hash. This way, the usual alternate decryption that is used in CCA-seurity can be provided till the time of challange encyption using this hash proof. Further, since this smooth projective hash is actually malleable, the Eval secret key sk_h need not include the key of this smooth projective hash.

Again, given a dual-system non-interactive proof Σ, consider the following algorithms for the more advanced KH-PKE scheme $\mathcal{R}$:

KeyGen: Generate $\mathbf{g}$ and a, k, f_1, f_2 randomly. Use sfK_1 of Σ to generate CRS σ and trapdoors τ and η with language parameters $\rho = (\mathbf{g}, \mathbf{g}^a)$. Set $pk = (\mathbf{g}, \mathbf{g}^a, \mathbf{g}^k, \mathbf{g}^{f_1}\mathbf{g}^{af_2}, \sigma)$, $\quad sk_h := \tau, \quad sk_d := (k, f_1, f_2)$.

Enc: Given plaintext m, generate $w \leftarrow \mathbb{Z}_q$ and compute (using P of Σ)
$c := (\mathbf{g}^w, \mathbf{g}^{aw}, \gamma, h, \mathsf{P}(\sigma, (\mathbf{g}^w, \mathbf{g}^{aw}), w, l))$, where $\gamma := m \cdot \mathbf{g}^{kw}$, $h = (\mathbf{g}^{f_1}\mathbf{g}^{af_2})^w$ and label $l = (\gamma, h)$.

Dec: Given ciphertext $c = (\rho, \hat{\rho}, \gamma, h, \pi)$, first check if $h = \rho^{f_1}\hat{\rho}^{f_2}$. Next, check if $\mathsf{V}(\sigma, \pi, (\rho, \hat{\rho}); (\gamma, h))$ of Σ holds, then compute $m := \gamma/\rho^k$.

Eval (multiplicative): Given ciphertexts $c_1 = (\rho_1, \hat{\rho}_1, \gamma_1, h_1, \pi_1)$ and $c_2 = (\rho_2, \hat{\rho}_2, \gamma_2, h_2, \pi_2)$, first check if $\mathsf{V}(\sigma, \pi_i, (\rho_i, \hat{\rho}_i); (\gamma_i, h_2))$ of Σ holds for al $i \in [1..2]$. Then compute: $\rho = \rho_1\rho_2\rho_3$, $\hat{\rho} = \hat{\rho}_1\hat{\rho}_2\hat{\rho}_3$, $\gamma = \gamma_1\gamma_2\gamma_3$, $h = h_1h_2h_3$ where $\langle \rho_3, \hat{\rho}_3, \gamma_3, h_3 \rangle$ is a fresh random tuple obtained by picking r at random and setting the tuple to be $\langle \mathbf{g}^r, (\mathbf{g}^a)^r, (\mathbf{g}^k)^r, (\mathbf{g}^{f_1}\mathbf{g}^{af_2})^r \rangle$. Then compute $\pi := \mathsf{sfSim}(\sigma, \tau, (\rho, \hat{\rho}), \beta = 1; (\gamma, h))$ using sfSim of Σ. Output ciphertext $c := (\rho, \hat{\rho}, \gamma, h, \pi)$.

Theorem 23 *The algorithms $\mathcal{R} = (KeyGen, Enc, Dec, Eval)$ above constitute a KH-CCA secure Keyed-Homomorphic Public Key Encryption scheme with multiplicative homomorphism, if Σ is a strong DSS-QA-NIZK for the parameterized Diffie-Hellman language (with language parameters distributed randomly). The Adversary's advantage in the KH-CCA security game is at most*

$$\begin{aligned}&(2 * L) * \text{ADV}^{pzk}(\Sigma, 1) \ + \text{ADV}^{pzk}(\Sigma, 2 * L) + \\ &O(L * (M + 2 * L)) * (\text{DIST}^{otzk}(\Sigma) + \text{ADV}^{pss}(\Sigma)) \ + (N + 1) * \text{ADV}_{\text{DDH}} + O(L/q).\end{aligned}$$

The proof is similar to the proof of Theorem 22, but in addition it allows alternate decryption till the time of generation of challenge ciphertext, even if the Eval key is revealed by using the smooth projective hash of the DH language provided by h.

D Proof of Realization of the UC-PAKE Functionality

In this section we state and prove that the protocol in Fig. 6.2.1 realizes the multi-session ideal functionality $\widehat{\mathcal{F}}_{\text{PAKE}}$, as long as the adversary does not corrupt a party such that its peer has already issued the key in a session and the adversary has not delivied the message from the peer to the party in that session.

Theorem 24 *Assuming the existence of SXDH-hard groups, the protocol in Fig 6.2.1 securely realizes the $\widehat{\mathcal{F}}_{\text{PAKE}}$ functionality in the $\mathcal{F}_{\text{CRS}}$ hybrid model, in the presence of adaptive corruption adversaries, , as long as an adversary does not corrupt a party after its peer has issued a key in a session and the adversary has not delivered the message in the session from the peer to the party.*

We start by defining the UC simulator in detail.

D.1 The Simulator for the UC Protocol.

We will assume that the adversary $\mathcal{A}$ in the UC protocol is dummy, and essentially passes back and forth commands and messages from the environment $\mathcal{Z}$. Thus, from now on we will use environment $\mathcal{Z}$ as the real adversary, which outputs a single bit. The simulator $\mathcal{S}$ will be the ideal world adversary for $\widehat{\mathcal{F}}_{\text{PAKE}}$. It is a universal simulator that uses $\mathcal{A}$ as a black box.

For each instance (session and a party), we will use subscript 2 along with a prime, to refer to variables received in the message from $\mathcal{Z}$ (i.e $\mathcal{A}$), and use subscript 1 to refer to variables computed in the instance under consideration. We will call a message *legitimate* if it was not altered by $\mathcal{Z}$, and delivered in the correct session and to the correct party.

The simulator $\mathcal{S}$ picks the CRS just as the semi-functional CRS generator sfK_1 picks the CRS for the DSS-QA-NIZK. This is statistically same as the real-world CRS in the UC protocol, except $\mathcal{S}$ sets $d = d_1 + a \cdot d_2$, and $e = e_1 + a \cdot e_2$, where d_1, d_2, e_1, e_2 are chosen randomly and independently from $\mathbb{Z}_q$. It retains $a, c, b, d_1, d_2, e_1, e_2, u_1, u_2$ as trapdoors. We will denote $u'_1 = u_1 b - d - ac$, and $u'_2 = u_2 b - e$.

The next main difference in the simulation of the real world parties is that $\mathcal{S}$ uses a dummy message μ instead of the real password which it does not have access to. Further, it decrypts the incoming message R'_2, S'_2, T'_2 to compute a pwd$'$, which it uses to call the ideal functionality's test function. If the test succeeds, it produces a sk (see below) and sends it to the ideal functionality to be output to the party concerned.

D.2 New Session: Sending a message to $\mathcal{Z}$.

On message ($\mathsf{NewSession}$, *sid*, ssid, i, j, role) from $\widehat{\mathcal{F}}_{\text{PAKE}}$, S starts simulating a new instance of the protocol Π for party P_i, peer P_j, session identifier ssid, and CRS set as above. We will denote this instance by (P_i, ssid). To simulate this instance, S chooses r_1, r'_1, r''_1, s_1 at random, and sets $R_1 = \mathbf{g}_1^{r_1}$, $S_1 = \mu \cdot \mathbf{a}^{r_1} \cdot \mathbf{g}_1^{r'_1}$, $\hat{\rho}_1 = \mathbf{b}^{s_1}$. Next, it computes $T_1 = \mathbf{g}_1^{r_1 \cdot (d_1 + \iota_1 e_1)} \cdot \mathbf{g}_1^{r''_1}$, where $\iota_1 = \mathcal{H}(sid, \mathsf{ssid}, P_i, P_j, R_1, S_1, \hat{\rho}_1)$ (note the use of μ instead of pwd). (Intuitively, this is what sfSim would compute for T_1 as well if membership-bit was set to false, which is the case here with high probability.)

It retains r_1, r'_1, r''_1, s_1 (and μ if chosen randomly). It then hands $R_1, S_1, T_1, \hat{\rho}_1$ to $\mathcal{Z}$ on behalf of this instance.

D.3 On Receiving a Message from $\mathcal{Z}$.

On receiving a message $R'_2, S'_2, T'_2, \hat{\rho}'_2$ from $\mathcal{Z}$ intended for this instance (P_i, ssid), the simulator S makes the real world protocol checks, namely group membership and non-triviality. If any of these checks fail, it issues a $\mathsf{TestPwd}$ call to $\widehat{\mathcal{F}}_{\text{PAKE}}$ with the dummy password μ, followed by a NewKey call with a random session key, which leads to the functionality issuing a random and independent session key to the party P_i (regardless of whether the instance was interrupted or compromised).

Otherwise, if the message received from $\mathcal{Z}$ is same as message sent by $\mathcal{S}$ on behalf of peer P_j in session ssid, then $\mathcal{S}$ just issues a NewKey call for P_i.

Else, it computes pwd$'$ by decrypting S'_2, i.e. setting it to $S'_2/(R'_2)^a$. $\mathcal{S}$ then calls $\widehat{\mathcal{F}}_{\text{PAKE}}$ with ($\mathsf{TestPwd}$, ssid, P_i, pwd$'$). Regardless of the reply from $\mathcal{F}$, it then issues a NewKey call for P_i with key computed as follows (recall, $R_1, S_1, \iota_1, r'_1, r''_1$ from earlier in this instance when the message was sent to $\mathcal{Z}$). Next,

$$\iota_2' = \mathcal{H}(sid, \mathsf{ssid}, P_j, P_i, R_2', S_2', \hat{\rho}_2'),\ \rho_1 = \mathbf{g}_2^{s_1},\ \theta_1 = \mathbf{c}^{s_1},\ \gamma_1 = (\mathbf{v}_1 \mathbf{v}_2^{\iota_2'})^{s_1},$$
$$W_1 = R_1^{(u_1' + \iota_1 u_2')/b} \cdot (S_1/\text{pwd}')^{c/b} \cdot T_1^{1/b}.$$

if $T_2' = (R_2')^{d+\iota_2' e}$ then set $\xi_1 = \mathbf{1}_T$ else set $\xi = \epsilon(\mathbf{g}_1, \mathbf{g}_2)^{s_1'}$, where s_1' is a fresh random value. Call $\widehat{\mathcal{F}}_{\text{PAKE}}$'s NewKey with session key

$$e(R_2', \gamma_1) \cdot e((R_2')^a, \theta_1) \cdot e(T_2', \rho_1) \cdot e(W_1, \hat{\rho}_2') \cdot \xi_1.$$

Note that this is how sfSim would compute W_1. By definition of $\widehat{\mathcal{F}}_{\text{PAKE}}$, this has the effect that if the pwd$'$ was same as the actual pwd previously recorded in $\widehat{\mathcal{F}}_{\text{PAKE}}$ (for this instance) then the session key is determined by the Simulator as above, otherwise the session key is set to a random and independent value.

D.4 Corruption

On receiving a $\mathsf{Corrupt}$ call from $\mathcal{Z}$ for instance P_i in session ssid, the simulator $\mathcal{S}$ calls the $\mathsf{Corrupt}$ routine of $\widehat{\mathcal{F}}_{\text{PAKE}}$ to obtain pwd. If $\mathcal{S}$ had already output a message to $\mathcal{Z}$, and not output sk_1 it computes

$$W_1 = R_1^{(u_1' + \iota_1 u_2')/b} \cdot (S_1/\text{pwd})^{c/b} \cdot T_1^{1/b}$$

and outputs this W_1 along with pwd, and ρ_1, θ_1, γ_1 as internal state of P_i. Note that this computation of W_1 is identical to the computation of W_1 in the computation of sk_1 (which is really output to $\mathcal{Z}$ only when pwd$'$ = pwd).

Without loss of generality, we can assume that in the real-world if the Adversary (or Environment $\mathcal{Z}$) corrupts an instance before the session key is output then the instance does not output any session key. This is so because the Adversary (or $\mathcal{Z}$) either sets the key for that session or can compute it from the internal state it broke into.

D.5 Proof of Indistinguishability - Series of Experiments.

We now describe a series of experiments between a probabilistic polynomial time challenger $\mathcal{C}$ and the environment $\mathcal{Z}$, starting with Expt_0 which we describe next. We will show that the view of $\mathcal{Z}$ in Expt_0 is same as its view in UC-PAKE ideal-world setting with $\mathcal{Z}$ interacting with $\widehat{\mathcal{F}}_{\text{PAKE}}$ and the UC-PAKE simulator $\mathcal{S}$ described above in Section D.1. We end with an experiment which is identical to the real world execution of the protocol in Fig 6.2.1. We will show that the environment has negligible advantage in distinguishing between these series of experiments, leading to a proof of realization of $\mathcal{F}_{\text{PAKE}}$ by the protocol Π.

Here is the complete code in Expt_0 (stated as it's overall experiment with $\mathcal{Z}$):

1. The challenger $\mathcal{C}$ picks the CRS just as in the real world, except it sets $d = d_1 + a \cdot d_2$, and $e = e_1 + a \cdot e_2$, where d_1, d_2, e_1, e_2 are chosen randomly and independently from $\mathbb{Z}_q$. It retains $a, c, b, d_1, d_2, e_1, e_2, u_1, u_2$ as trapdoors. Also, denote $u_1' = u_1 b - d - ac$, and $u_2' = u_2 b - e$.

2. On receiving $\texttt{NewSession}, sid, \mathsf{ssid}, P_i, P_j, \text{pwd}, role$ from $\mathcal{Z}$, $\mathcal{C}$ generates R_1, S_1, T_1, $\hat{\rho}_1 1$ by choosing r_1, r_1', r_1'', s_1 at random, and setting $R_1 = \mathbf{g}_1^{r_1}$, $S_1 = \mu \cdot \mathbf{a}^{r_1} \cdot \mathbf{g}_1^{r_1'}$, $T_1 = \mathbf{g}_1^{r_1 \cdot (d_1 + \iota_1 e_1)} \cdot \mathbf{g}_1^{r_1''}$, $\hat{\rho}_1 = \mathbf{b}^{s_1}$. It sends these values to $\mathcal{Z}$.

3. On receiving $R_2', S_2', T_2', \hat{\rho}_2'$ from $\mathcal{Z}$, intended for session ssid and party P_i (and assuming no corruption of this instance)

   (a) if the received elements are either not in their respective groups, or are trivially 1, output sk_1 chosen randomly and independently from $\mathbb{G}_T$.

   (b) Otherwise, if the message received is identical to message sent by $\mathcal{C}$ in the same session (i.e. same ssid) on behalf of the peer, then output $\mathrm{sk}_1 \xleftarrow{\$} \mathbb{G}_T$ (unless the simulation of peer also received a legitimate message and its key has already been set, in which case the same key is used to output sk_1 here).

   (c) Else, compute $\mathrm{pwd}' = S_2'/(R_2')^a$. If $\mathrm{pwd}' \neq \mathrm{pwd}$ (note pwd was given in `NewSession` request), then output sk_1 randomly and independently from $\mathbb{G}_T$.

   (d) Else, compute $\iota_2' = \mathcal{H}(sid, \mathsf{ssid}, P_j, P_i, R_2', S_2', \hat{\rho}_2')$, $\rho_1 = \mathbf{g}_2^{s_1}$, $\theta_1 = \mathbf{c}^{s_1}$, $\gamma_1 = (\mathbf{v}_1\mathbf{v}_2^{\iota_2'})^{s_1}$, $W_1 = R_1^{(u_1'+\iota_1 u_2')/b} \cdot (S_1/\mathrm{pwd})^{c/b} \cdot T_1^{1/b}$. if $(T_2' \neq (R_2')^{d+\iota_2' e})$ then output a random value in $\mathbb{G}_T$ else Output

$$e(R_2', \gamma_1) \cdot e(S_2'/\mathrm{pwd}, \theta_1) \cdot e(T_2', \rho_1) \cdot e(W_1, \hat{\rho}_2').$$

4. On a $\mathsf{Corrupt}$ call, if step 2 has already happened then output s_1, pwd and $W_1 = R_1^{(u_1'+\iota_1 u_2')/b} \cdot (S_1/\mathrm{pwd})^{c/b} \cdot T_1^{1/b}$.

All outputs of sk_1 are also accompanied with sid, ssid (but are not mentioned above for ease of exposition).

Note that each instance has two asynchronous phases: a phase in which $\mathcal{C}$ outputs $R_1, S_1, ...$ to $\mathcal{Z}$, and a phase where it receives a message from $\mathcal{Z}$. However, $\mathcal{C}$ cannot output sk_1 until it has completed both phases. These orderings are dictated by $\mathcal{Z}$. We will consider two different kinds of temporal orderings. A temporal ordering of different instances based on the order in which $\mathcal{C}$ outputs sk_1 in an instance will be called **temporal ordering by key output**. A temporal ordering of different instances based on the order in which $\mathcal{C}$ outputs its first message (i.e. $R_1, S_1, ...$) will be called **temporal ordering by message output**. It is easy to see that $\mathcal{C}$ can dynamically compute both these orderings by maintaining a counter (for each ordering).

It is straightforward to inspect that the view of $\mathcal{Z}$ in Expt_0 is identical to its view in its combined interaction with $\widehat{\mathcal{F}}_{\text{PAKE}}$ and $\mathcal{S}$, as $\mathcal{C}$ has just combined the code of $\widehat{\mathcal{F}}_{\text{PAKE}}$ and $\mathcal{S}$ (noting that in step 3(d), $\mathrm{pwd} = \mathrm{pwd}'$)

Expt_1 : In this experiment step 3(c) is dropped altogether and step 3(d) is altered as follows: In step 3(d) in Expt_0, the condition $T_2' \neq (R_2')^{d+\iota_2' e}$ is replaced by if $(S_2' \neq \mathrm{pwd} \cdot (R_2')^a)$ or $(T_2' \neq (R_2')^{d+\iota_2' e})$. Rest of the computation of sk_1 in step 3(d) remains the same.

We claim that the view of $\mathcal{Z}$ is statistically identical in Expt_0 and Expt_1. This follows by noting that $S_2' \neq \mathrm{pwd} \cdot (R_2')^a$ is equivalent to the condition $\mathrm{pwd}' \neq \mathrm{pwd}$ in Expt_0. The condition $S_2' = \mathrm{pwd} \cdot (R_2')^a$ held in step 3(d) in Expt_0, as that step was only reached if this condition held.

At this point, Expt_1 can equivalently be written as follows:

1. The challenger $\mathcal{C}$ picks the CRS just as in the real world, except it sets $d = d_1 + a \cdot d_2$, and $e = e_1 + a \cdot e_2$, where d_1, d_2, e_1, e_2 are chosen randomly and independently from $\mathbb{Z}_q$. It retains $a, c, b, d_1, d_2, e_1, e_2, u_1, u_2$ as trapdoors.

2. On receiving $\texttt{NewSession}, sid, \mathsf{ssid}, P_i, P_j, \mathrm{pwd}, role$ from $\mathcal{Z}$, $\mathcal{C}$ generates R_1, S_1, T_1, $\hat{\rho}_1 1$ by choosing r_1, r_1', r_1'', s_1 at random, and setting $R_1 = \mathbf{g}_1^{r_1}$, $S_1 = \mu \cdot \mathbf{a}^{r_1} \cdot \mathbf{g}_1^{r_1'}$, $T_1 = \mathbf{g}_1^{r_1 \cdot (d_1 + \iota_1 e_1)} \cdot \mathbf{g}_1^{r_1''}$, $\hat{\rho}_1 = \mathbf{b}^{s_1}$. It sends these values to $\mathcal{Z}$.

3. On receiving $R_2', S_2', T_2', \hat{\rho}_2'$ from $\mathcal{Z}$(and assuming no corruption of this instance),

   (a) if the received elements are either not in their respective groups, or are trivially 1, output sk_1 chosen randomly and independently from $\mathbb{G}_T$.

   (b) Otherwise, if the message received is identical to message sent by $\mathcal{C}$ in the same session (i.e. same ssid) on behalf of the peer, then output $\mathrm{sk}_1 \xleftarrow{\$} \mathbb{G}_T$ (unless the simulation of peer also received a legitimate message and its key has already been set, in which case the same key is used to output sk_1 here).

   (c) -

   (d) Else, compute $\iota_2' = \mathcal{H}(sid, \mathsf{ssid},\ P_j, P_i,\ R_2', S_2', \hat{\rho}_2')$, $\rho_1 = \mathbf{g}_2^{s_1}$, $\theta_1 = \mathbf{c}^{s_1}$, $\gamma_1 = (\mathbf{v}_1 \mathbf{v}_2^{\iota_2'})^{s_1}$, $W_1 = R_1^{(u_1' + \iota_1 u_2')/b} \cdot (S_1/\mathrm{pwd})^{c/b} \cdot T_1^{1/b}$. if $(S_2' \neq \mathrm{pwd} \cdot (R_2')^a)$ or $(T_2' = (R_2')^{d + \iota_2' e})$ then output a random value in $\mathbb{G}_T$ else Output

$$e(R_2', \gamma_1) \cdot e(S_2'/\mathrm{pwd}, \theta_1) \cdot e(T_2', \rho_1) \cdot e(W_1, \hat{\rho}_2').$$

4. On a $\mathsf{Corrupt}$ call, if step 2 has already happened then output s_1, pwd and $W_1 = R_1^{(u_1' + \iota_1 u_2')/b} \cdot (S_1/\mathrm{pwd})^{c/b} \cdot T_1^{1/b}$.

Expt_2 : In this experiment the challenger in step 2 computes S_1 in each instance as $\mathrm{pwd} \cdot \mathbf{a}^{r_1} \cdot \mathbf{g}_1^{r_1'}$. Note the use of pwd instead of μ.

This is statistically the same, as in each instance the challenger picks a fresh and random r_1', and it is not used anywhere else.

Expt_3 : In each instance T_1 is computed as follows: $T_1 = \mathbf{g}_1^{r_1 \cdot (d + \iota_1 e)} \cdot \mathbf{g}_1^{r_1''}$.

This is statistically identical as T_1 has a random factor $\mathbf{g}_1^{r_1''}$.

Expt_4 : In this experiment the challenger $\mathcal{C}$ generates the CRS slightly differently. It picks u_1' and u_2' at random, and sets $u_1 = (u_1' + d + c \cdot a)/b$, and $u_2 = (u_2' + e)/b$. Rest of the CRS computation remains the same. Note that a is not used anymore in the computation of group $\mathbb{G}_2$ CRS elements, in particular $\mathbf{v}_1$.

The view of the adversary in Expt_3 and Expt_4 is statistically the same.

Expt_5 : In each instance S_1 is computed as follows: $\mathrm{pwd} \cdot \mathbf{a}^{r_1}$. Further, T_1 is computed as follows: $T_1 = R_1^{d + \iota_1 e}$. Further, later on in the instance, on receiving a message from $\mathcal{Z}$, W_1 is computed as $W_1 = R_1^{(u_1 + \iota_1 u_2)}$ (also the same change in Corrupt; actually this change is just syntactic and follows from the change in computation of S_1 and T_1). In other words, the computation of S_1, T_1 and W_1 is as in the real-world (this can be seen from the definition of $\mathbf{w}_1$ and $\mathbf{w}_2$ in the CRS).

To show that Expt_4 is computationally indistinguishable from Expt_5, we define several hybrid experiments $\mathsf{Expt}_{4,i}$ inductively. Experiment $\mathsf{Expt}_{4,0}$ is identical to Expt_4. If there are a total of N instances, $\mathsf{Expt}_{4,N}$ will be identical to Expt_5. Experiment $\mathsf{Expt}_{4,1+1}$ differs from experiment $\mathsf{Expt}_{4,i}$ in only (temporally ordered by message output) the $(i+1)$-th instance. While in $\mathsf{Expt}_{4,i}$, the $(i+1)$-th instance is simulated by $\mathcal{C}$ as in Expt_4, in $\mathsf{Expt}_{4,i+1}$ this instance is simulated as in Expt_5.

Lemma 25 *For all $i : 0 \leq i \leq N$, the view of $\mathcal{Z}$ in experiment $\mathsf{Expt}_{4,i+1}$ is computationally indistinguishable from the view of $\mathcal{Z}$ in $\mathsf{Expt}_{4,i}$.*

Proof: The proof will follow the proof of one-time full ZK in Appendix B, except that we will also employ DDH to replace the fake tuples with real tuples.

We define several hybrid experiments. Experiment $\mathbf{G}_0$ is identical to $\mathsf{Expt}_{4,i}$.

In experiment $\mathbf{G}_1$, the CRS is picked differently by $\mathcal{C}$. instead of picking d_1, d_2 at random (and similarly e_2, e_2 at random), picks d and d_2 randomly and independently and sets $d_1 = d - a \cdot d_2$ (and $e_1 = e - a \cdot e_2$). Rest of the CRS computation remains the same, and it retains a, c, b, d, d_2, e, e_2, u'_1, u'_2. This is statistically the same distribution.

In $\mathbf{G}_2$, in the $(i+1)$-th instance T is computed differently:

$$T = R_1^{d+\iota_1 e - a\cdot(d_2+\iota_1 e_2)} \cdot (S_1/\mathrm{pwd})^{d_2+\iota_1 e_2} \tag{3}$$

This is statistically the same as d_2 was chosen randomly and independently of d. All other instances are only using d and e in their computation. Thus, the r''_1 in this instance can be replaced by $(d_2 + \iota_1 e_2) r'_1$ (note S_1 is computed as $\mathrm{pwd} \cdot \mathbf{a}^{r_1} \cdot \mathbf{g}_1^{r'_1}$ in previous experiments).

In the next experiment $\mathbf{G}_3$, in step 3(d) of each instance, the condition "if $(S'_2 \neq \mathrm{pwd} \cdot (R'_2)^a)$ or $(T'_2 = (R'_2)^{d+\iota'_2 e})$" is replaced by "if $(T'_2 \neq (R'_2)^{d_1+\iota'_2 e_1} \cdot (S'_2/\mathrm{pwd})^{d_2+\iota'_2 e_2})$". Rest of the computation of sk_1 remains the same. This is the same as using sfV instead of pV, and the proof of $\mathbf{G}_3$ being statistically indistinguishable from $\mathbf{G}_2$ is exactly the same as in proof of $\mathbf{H}_1$ and $\mathbf{H}_2$ in one-time full-ZK property proof of theorem 6 (see lemma 7) noting that $sid, \mathsf{ssid}, P, P_j$ and $\hat{\rho}$ are used as label. Note that the proof of statistical indistinguishability of $\mathbf{H}_1$ and $\mathbf{H}_2$ itself requires a hybrid argument.

In the next experiment $\mathbf{G}_4$, the computation of each of the instances (except the $(i+1)$-th instance undergoes the following change: in the j-th instance, if $j < (i+1)$, then set $r''_1 = 0$, else choose r''_1 at random. Compute $T_1 = R_1^{d_1+\iota_1 e_1} \cdot (S_1/\mathrm{pwd})^{d_2+\iota_1 e_2} \cdot \mathbf{g}_1^{r''_1}$.

Experiments $\mathbf{G}_4$ and $\mathbf{G}_3$ are statistically identical, and the proof is same as for the proof of indistinguishability of Expt_2 and Expt_3.

In the next experiment $\mathbf{G}_5$, the challenger produces the rest of the CRS using $\rho = \mathbf{g}_1, \mathbf{g}_1^a$, where a is chosen randomly from $\mathbb{Z}_q$. In particular, it chooses $d_1, d_2, e_1, e_2, u'_1, u'_2, b, c$ at random, and defines the CRS using ρ and these values. This is statistically the same.

In the next experiment $\mathbf{G}_6$, the challenger generates the S_1 in the $(i+1)$-th instance as follows: $S_1 = \mathrm{pwd} \cdot \mathbf{a}^{r_1}$. That the view of $\mathcal{Z}$ in experiments $\mathbf{G}_5$ and $\mathbf{G}_6$ are computationally indistinguishable follows from the DDH assumption in group $\mathbb{G}_1$.

Now, we unwind our way back to $\mathsf{Expt}_{4,i+1}$ following all of the above hybrid games back-wards, and that completes the proof. $\square$ $\square$

At this point, Expt_5 can be described as follows:

1. The challenger $\mathcal{C}$ picks the CRS just as in the real world, except it sets $d = d_1 + a \cdot d_2$, and $e = e_1 + a \cdot e_2$, where d_1, d_2, e_1, e_2 are chosen randomly and independently from $\mathbb{Z}_q$. It retains $a, c, b, d_1, d_2, e_1, e_2, u_1, u_2$ as trapdoors.

2. On receiving $\texttt{NewSession}, sid, \mathsf{ssid}, P_i, P_j, \mathrm{pwd}, role$ from $\mathcal{Z}$, $\mathcal{C}$ generates R_1, S_1, T_1, $\hat{\rho}_1$ by choosing r_1, s_1 at random, and setting $R_1 = \mathbf{g}_1^{r_1}$, $S_1 = \mathrm{pwd} \cdot \mathbf{a}^{r_1}$, $T_1 = (\mathbf{d}\mathbf{e}^{\iota_1})^{r_1}$, $\hat{\rho}_1 = \mathbf{b}^{s_1}$. It sends these values to $\mathcal{Z}$.

3. On receiving $R_2', S_2', T_2', \hat{\rho}_2'$ from $\mathcal{Z}$(and assuming no corruption of this instance),

   (a) if the received elements are either not in their respective groups, or are trivially 1, output sk_1 chosen randomly and independently from $\mathbb{G}_T$.

   (b) Otherwise, if the message received is identical to message sent by $\mathcal{C}$ in the same session (i.e. same ssid) on behalf of the peer, then output $\mathrm{sk}_1 \overset{\$}{\leftarrow} \mathbb{G}_T$ (unless the simulation of peer also received a legitimate message and its key has already been set, in which case the same key is used to output sk_1 here).

   (c) -

   (d) Else, compute $\iota_2' = \mathcal{H}(sid, \mathsf{ssid},\ P_j, P_i,\ R_2', S_2', \hat{\rho}_2')$, if $(S_2' \neq \mathrm{pwd} \cdot (R_2')^a)$ or $(T_2' \neq (R_2')^{d+\iota_2' e})$ then output a random value in $\mathbb{G}_T$,

   (e) else, compute $\rho_1 = \mathbf{g}_2^{s_1}$, $\theta_1 = \mathbf{c}^{s_1}$, $\gamma_1 = (\mathbf{v}_1\mathbf{v}_2^{\iota_2'})^{s_1}$, $W_1 = (\mathbf{w}_1\mathbf{w}_2^{\iota_2})^{r_1}$, and output

$$e(R_2', \gamma_1) \cdot e(S_2'/\mathrm{pwd}, \theta_1) \cdot e(T_2', \rho_1) \cdot e(W_1, \hat{\rho}_2').$$

4. On a $\mathsf{Corrupt}$ call, if step 2 has already happened then output s_1, pwd and $W_1 = (\mathbf{w}_1\mathbf{w}_2^{\iota_2})^{r_1}$

D.5.1 Handling Legitimate Messages

Expt_6 : In this experiment the step 3(b) is modified as follows:
Step 3(b): Otherwise, if the message received is identical to message sent by $\mathcal{C}$ in the same session (i.e. same SSID) on behalf of the peer, **and** if simulation of peer also received a legitimate message and its key has already been set, then output that same key here. Else, go to step 3(e).

To show that Expt_6 is indistinguishable from Expt_5 we need to go through several hybrid experiments. In each subsequent hybrid experiment one more session (and possibly its peer) are modified, and the order in which these sessions are handled is determined by temporal order of message sending. In the hybrid experiment $\mathsf{Expt}_{5,i}$, the step 3(b) in the peer of the i-th temporally ordered session is modified as required in Expt_6 description above. Experiment $\mathsf{Expt}_{5,0}$ is same as experiment Expt_5, and experiment $\mathsf{Expt}_{5,N}$ is same as experiment $\mathsf{Expt}7$.

Lemma 26 *For all $i \in [1..N]$, experiment $\mathsf{Expt}_{5,i}$ is computationally indistinguishable from $\mathsf{Expt}_{5,i-1}$.*

Proof: The lemma is proved using several hybrid experiments of its own. The experiment $\mathbf{H}_0$ is same as $\mathsf{Expt}_{5,i-1}$.

In experiment $\mathbf{H}_1$ the CRS is set by choosing u_1', u_2' at random and setting $u_1 = (u_1' + d + ca)/b$ and $u_2 = (u_2' + e)/b$. Thus, a is not used in generation of $\mathbb{G}_2$ elements.

In experiment $\mathbf{H}_2$, in session i, the value W_1 (in step 3(e) or corruption) is generated using R_1, S_1/pwd and T_1 using the trapdoors (as in Expt_0), i.e. $R_1^{(u_1'+\iota_1 u_2')/b} \cdot (S_1/\mathrm{pwd})^{c/b} \cdot T_1^{1/b}$.

In experiment $\mathbf{H}_3$, in step 3(d) of every session, the condition "if $(S_2' \neq \mathrm{pwd} \cdot (R_2')^a)$ or $(T_2' \neq (R_2')^{d+\iota_2' e})$" is replaced by "if $(T_2' \neq (R_2')^{d_1+\iota_2' e_1} \cdot (S_2'/\mathrm{pwd})^{d_2+\iota_2' e_2})$". This is the same as using sfV

instead of pV, and the proof of $\mathbf{G}_3$ being statistically indistinguishable from $\mathbf{G}_2$ is exactly the same as in proof of $\mathbf{H}_1$ and $\mathbf{H}_2$ in one-time full-ZK property proof of theorem 6 (see lemma 7) noting that $sid, \mathsf{ssid}, P_, P_j$ and $\hat{\rho}$ are used as label. Note that the proof of statistical indistinguishability of $\mathbf{H}_1$ and $\mathbf{H}_2$ itself requires a hybrid argument.

In experiment $\mathbf{H}_4$, in session i, the values R_1, S_1 and T_1 are generated as $R_1 = \mathbf{g}_1^{r_1}$, $S_1 = \mathrm{pwd} \cdot \mathbf{a}^{r_1} \cdot \mathbf{g}_1^{r'_1}$ and $T_1 = \mathbf{g}_1^{r_1 \cdot (d+\iota_1 e)} \cdot \mathbf{g}_1^{(d_2+\iota_1 e_2) r_1}$, where r_1, r'_1 are random and independent. This follows by employing DDH on $\mathbf{g}_1, \mathbf{g}_1^{r_1}, \mathbf{a}$ and either $\mathbf{g}_1^{ar_1}$ or $\mathbf{g}_1^{ar_1+r'_1}$.

In experiment $\mathbf{H}_5$, the CRS is set as follows. The values u_1 and u_2 are chosen at random and set $u'_1 = u_1 b - d - ca$ and $u'_2 = u_2 b - e$. This is statistically the same as $\mathbf{H}_4$. Moreover, in every step 3(d), the condition is changed back to as in $\mathbf{H}_2$. Further, in peer of session i, in step 3(e), the value $e(R'_2, \gamma_1) \cdot e(S'_2/\mathrm{pwd}, \theta_1) \cdot e(T'_2, \rho_1)$ is instead computed as $e((R'_2)^{u_1+\iota'_2 u_2}, \mathbf{g}_2^{bs_1})$, as the message received is in the language.

In experiment $\mathbf{H}_6$, in session i, change step 3(b) as follows: Step 3(b): Otherwise, if the message received is identical to message sent by $\mathcal{C}$ in the same session (i.e. same SSID) on behalf of the peer,

- if simulation of peer also received a legitimate message and its key has already been set, then output that same key here. If peer is corrupted, output the key supplied by the Adversary.

- Else, compute $\iota'_2 = \mathcal{H}(sid, \mathsf{ssid},\ P_j, P_i,\ R'_2, S'_2, \hat{\rho}'_2)$, $\rho_1 = \mathbf{g}_2^{s_1}$, $\theta_1 = \mathbf{c}^{s_1}$, $\gamma_1 = (\mathbf{v}_1 \mathbf{v}_2^{\iota'_2})^{s_1}$, $W_1 = R_1^{(u'_1+\iota_1 u'_2)/b} \cdot (S_1/\mathrm{pwd})^{c/b} \cdot T_1^{1/b}$. Pick c'_2, s'_2 at random and set $\theta_1 = e(T_1 \cdot (S_1/\mathrm{pwd})^{c'_2}, \mathbf{g}_2)^{s'_2} \cdot e(R_1, \mathbf{g}_2)^{-(d+\iota_1 e + c'_2 \cdot a) s'_2}$. Output
$$e(R'_2, \gamma_1) \cdot e(S'_2/\mathrm{pwd}, \theta_1) \cdot e(T'_2, \rho_1) \cdot e(W_1, \hat{\rho}'_2) \cdot \theta_1.$$

(note: in all other cases step 3(d) and (e) are still being performed).

The experiments $\mathbf{H}_6$ and $\mathbf{H}_5$ are statistically indistinguishable by a simple information theoretic argument, since $S_1/\mathrm{pwd} \neq R_1^a$ (see experiment $\mathbf{H}_4$), and c'_2 is chosen randomly, and thus θ_1 is random in target group $\mathbb{G}_T$.

In experiment $\mathbf{H}_7$, in the above step 3(b) in session i, the term $e(W_1, \hat{\rho}'_2)$ is replaced by $e(R_1^{(u'_1+\iota_1 u'_2)} \cdot (S_1/\mathrm{pwd})^c \cdot T_1, \mathbf{g}_2^{s_2})$, where s_2 is the value chosen in the peer of session i (note session i in step 3(b) received the message exactly as generated in its peer, and hence $\hat{\rho}'_2 = \mathbf{g}_2^{bs_2}$.

In experiment $\mathbf{H}_8$, in above step 3(b) of session i, in computation of the term θ_1, the value c'_2 is replaced by c itself. Experiments $\mathbf{H}_8$ and $\mathbf{H}_7$ are indistinguishable follows by employing DDH on $\mathbf{g}_2, \mathbf{g}_2^{s'_2}, \mathbf{c}$, and either $\mathbf{g}_2^{cs'_2}$ (real DDH challenge) or $\mathbf{g}_2^{c'_2 s'_2}$ (fake DDH challenge).

In experiment $\mathbf{H}_9$, the term θ_1 is not computed at all, and is dropped from the computation of the session key output. Again, experiments $\mathbf{H}_9$ and $\mathbf{H}_8$ are indistinguishable by employing DDH on tuples $\mathbf{g}_2, \mathbf{g}_2^b, \mathbf{g}_2^{bs_2}$ and either $\mathbf{g}_2^{s_2}$ (yielding $\mathbf{H}_9$) or $\mathbf{g}_2^{s_2+s'_2}$ (yielding $\mathbf{H}_8$). Note that $u'_1 = u_1 b - d - ca$ and $u'_2 = u_2 b - e$. Note that in session i, in step 3(e), computation of W_1 uses $1/b$. However, step 3(e) is identical in both $\mathbf{H}_9$ and $\mathbf{H}_8$. Thus, probability for a adversary to distinguish between $\mathbf{H}_8$ and $\mathbf{H}_9$ remains upper bounded by maximum distinguishing probability of DDH challenges.

In experiment $\mathbf{H}_{10}$, in step 3(b) of session i, the term $e(R_1^{(u'_1+\iota_1 u'_2)} \cdot (S_1/\mathrm{pwd})^c \cdot T_1, \mathbf{g}_2^{s_2})$ is replaced back by $e(W_1, \hat{\rho}'_2)$. Experiments $\mathbf{H}_{10}$ and $\mathbf{H}_9$ are identical by the same argument as employed for experiments $\mathbf{H}_6$ and $\mathbf{H}_7$.

In experiment $\mathbf{H}_{11}$, the CRS is set by choosing u_1', u_2' at random and setting $u_1 = (u_1' + d + ca)/b$ and $u_2 = (u_2' + e)/b$. Thus, a is not used in generation of $\mathbb{G}_2$ elements. Further, in evert step 3(d) the condition is changed to as in $\mathbf{H}_3$.

In experiment $\mathbf{H}_{12}$, in session i, R_1, S_1 and T_1 are generated as $R_1 = \mathbf{g}^{r_1}$, $S_1 = \text{pwd}\mathbf{a}^{r_1}$, and $T_1 = (\mathbf{d}\mathbf{e}^{\iota_1})^{r_1}$, by employing DDH just as in experiments Expt_3 and Expt_2.

In experiment $\mathbf{H}_{13}$, in session i, in step 3(b), 3(e) and corruption step, W_1 is computed as $(\mathbf{w}_1\mathbf{w}_2^{\iota_1})^{r_1}$. This is statistically the same, as R_1, S_1 and T_1 are being correctly generated now.

In experiment $\mathbf{H}_{14}$, in step 3(d) of every session, the condition "if $(T_2' \neq (R_2')^{d_1+\iota_2' e_1} \cdot (S_2'/\text{pwd})^{d_2+\iota_2' e_2})$" is replaced back by "if $(S_2' \neq \text{pwd} \cdot (R_2')^a)$ or $(T_2' \neq (R_2')^{d+\iota_2' e})$". This is statistically the same for efficieny adversaries as shown above for Expt_2 and Expt_1.

In experiment $\mathbf{H}_{15}$, the CRS is set as follows. The values u_1 and u_2 are chosen at random and set $u_1' = u_1 b - d - ca$ and $u_2' = u_2 b - e$. □

At this point, Expt_6 can be described as follows:

1. The challenger $\mathcal{C}$ picks the CRS just as in the real world, except it sets $d = d_1 + a \cdot d_2$, and $e = e_1 + a \cdot e_2$, where d_1, d_2, e_1, e_2 are chosen randomly and independently from $\mathbb{Z}_q$. It retains $a, c, b, d_1, d_2, e_1, e_2, u_1, u_2$ as trapdoors.

2. On receiving $\texttt{NewSession}, sid, \mathsf{ssid}, P_i, P_j, \text{pwd}, role$ from $\mathcal{Z}$, $\mathcal{C}$ generates R_1, S_1, T_1, $\hat{\rho}_1$ by choosing r_1, s_1 at random, and setting $R_1 = \mathbf{g}_1^{r_1}$, $S_1 = \text{pwd} \cdot \mathbf{a}^{r_1}$, $T_1 = (\mathbf{d}\mathbf{e}^{\iota_1})^{r_1}$, $\hat{\rho}_1 = \mathbf{b}^{s_1}$. It sends these values to $\mathcal{Z}$.

3. On receiving $R_2', S_2', T_2', \hat{\rho}_2'$ from $\mathcal{Z}$(and assuming no corruption of this instance) compute $\iota_2' = \mathcal{H}(sid, \mathsf{ssid}, P_j, P_i, R_2', S_2', \hat{\rho}_2')$, and

   (a) if the received elements are either not in their respective groups, or are trivially 1, output sk_1 chosen randomly and independently from $\mathbb{G}_T$.

   (b) Otherwise, if the message received is identical to message sent by $\mathcal{C}$ in the same session (i.e. same SSID) on behalf of the peer, **and** if simulation of peer also received a legitimate message and its key has already been set, then output that same key here. Else, let $\xi_1 = \mathbf{1}_T$ and go to step 3(e).

   (c) -

   (d) Else, if $(S_2' \neq \text{pwd} \cdot (R_2')^a)$ or $(T_2' \neq (R_2')^{d+\iota_2' e})$ then let ξ_1 be a fresh random value from $\mathbb{G}_T$ else let $\xi_1 = \mathbf{1}_T$. Go to step 3(e).

   (e) compute $\rho_1 = \mathbf{g}_2^{s_1}$, $\theta_1 = \mathbf{c}^{s_1}$, $\gamma_1 = (\mathbf{v}_1\mathbf{v}_2^{\iota_2'})^{s_1}$, $W_1 = (\mathbf{w}_1\mathbf{w}_2^{\iota_2})^{r_1}$, and output

$$e(R_2', \gamma_1) \cdot e(S_2'/\text{pwd}, \theta_1) \cdot e(T_2', \rho_1) \cdot e(W_1, \hat{\rho}_2') \cdot \xi_1.$$

4. On a $\mathsf{Corrupt}$ call, if step 2 has already happened then output s_1, pwd and $W_1 = (\mathbf{w}_1\mathbf{w}_2^{\iota_2})^{r_1}$

D.5.2 Handling Adversarial Messages

Expt_7 : In this experiment in step 3(d), in each instance $\mathcal{C}$ picks random values $c'_1, s'_1 \in \mathbb{Z}_q$m and computes ξ_1 differently as follows (all other values are computed as in Expt_6):
$\xi_1 = e(T'_2 \cdot (S'_2/\mathrm{pwd})^{c'_1}, \mathbf{g}_2)^{s'_1} \cdot e(R'_2, \mathbf{g}_2)^{-(d+\iota'_2 e + c'_1 \cdot a)s'_1}$.

This computation of ξ_1 in step 3(d) is statistically indistinguishable by a simple information-theoretic argument.

Expt_8 : In this experiment, in each instance ξ_1 is computed without using a fresh c'_1, but instead using c itself, i.e. $\xi_1 = e(T'_2 \cdot (S'_2/\mathrm{pwd})^{c}, \mathbf{g}_2)^{s'_1} \cdot e(R'_2, \mathbf{g}_2)^{-(d+\iota'_2 e + ca)s'_1}$.

A standard hybrid argument. employing DDH assumption in group $\mathbb{G}_2$ on $\mathbf{g}_2$, $\mathbf{g}_2^c$, $\mathbf{g}_2^{s'_1}$ and either $\mathbf{g}_2^{c \cdot s'_1}$ (real DDH tuple) or $\mathbf{g}_2^{c'_1 \cdot s'_1}$ (fake DDH tuple), shows that Expt_7 and Expt_8 are computationally indistinguishable by $\mathcal{Z}$.

Expt_9 : In this experiment, in each instance ξ_1 is not computed at all and is not used as a factor in computation of sk_1 in step 3(e). Thus, the step 3(d) does not exist in this experiment.

First note that in Expt_8, sk_1 can be equivalently computed by $\mathcal{C}$ as follows:
$\rho_1 = \mathbf{g}_2^{s_1+s'_1}$, $\theta_1 = \mathbf{c}^{s_1+s'_1}$, $\gamma_1 = (\mathbf{v}_1 \mathbf{v}_2^{\iota'_2})^{s_1} \cdot \mathbf{g}_2^{-(d+\iota'_2 e+ca)s'_1}$, $W_1 = (\mathbf{w}_1 \mathbf{w}_2^{\iota_1})^{r_1}$,
$\mathrm{sk}_1 = e(T'_2, \rho_1) \cdot e(\frac{S'_2}{\mathrm{pwd}}, \theta_1) \cdot e(R'_2, \gamma_1) \cdot e(W_1, \hat{\rho}'_2)$. Note that if the message reseived was legitimate then ($S'_2 = \mathrm{pwd} \cdot (R'_2)^a$) and ($T'_2 = (R'_2)^{d+\iota'_2 e}$). We also remark that if the instance has already been corrupted (and s_1 disclosed to $\mathcal{Z}$) then this step never happens. Now, it is an easy matter to show that Expt_9 and Expt_8 are computationally indistinguishable by employing DDH assumption (multiple times using a hybrid argument) on $\mathbf{g}_1$, $\mathbf{g}_2^b$, $\mathbf{g}_2^{bs_1}$ and either $\mathbf{g}_2^{s_1}$ (real DDH challenge yielding experiment Expt_{13}) or $\mathbf{g}_2^{s_1+s'_1}$ (fake DDH challenge yielding experiment Expt_{12}).

At this point, i.e. in Expt_9, the computation of step 3(d) is as follows:
Step 3(d): Compute $\rho_1 = \mathbf{g}_2^{s_1}$, $\theta_1 = \mathbf{c}^{s_1}$, $\gamma_1 = (\mathbf{v}_1 \mathbf{v}_2^{\iota'_2})^{s_1}$, $W_1 = (\mathbf{w}_1 \mathbf{w}_2^{\iota_1})^{r_1}$ and output $\mathrm{sk}_1 = e(T'_2, \rho_1) \cdot e(S'_2, \theta_1) \cdot e(R'_2, \gamma_1) \cdot e(W_1, \hat{\rho}'_2)$.

Expt_{10} : In this experiment the step 3(b) is dropped. In other words, the challenger code goes straight from 3(a) to 3(e).

Experiments Expt_{10} and Expt_{15} produce the same view for $\mathcal{Z}$, since if both peers (of a session) received legitimate messages forwarded by $\mathcal{Z}$, then step 3(e) computes the same session key in both instances.

Expt_{11} : In this experiment, the challenger chooses d and e randomly and uniformly, i.e. without first choosing d_1, d_2, e_1, e_2 and then computing d and e in terms of these and other variables.

It is straightforward to see that the view of $\mathcal{Z}$ is same in Expt_{10} and Expt_{11}.

Finally, a simple examination shows that the view of $\mathcal{Z}$ in Expt_{11} is identical to the real world protocol.

E Proof of Realization of the Non-information Oracle based UC-RPAKE

Theorem 10 Assuming the existence of SXDH-hard groups, there exists a multi-session non-information oracle $\hat{\mathcal{N}}$ such that the protocol in Fig 6.2.1 securely realizes the $\widehat{\mathcal{F}}^{\hat{\mathcal{N}}}_{\text{RPAKE}}$ functionality in the $\mathcal{F}_{\text{CRS}}$ hybrid model, in the presence of adaptive corruption adversaries.

We now describe the changes needed to the proof of section D. The multi-session non-information oracle $\hat{\mathcal{N}}$ works as follows. In the first phase it expects from the adversary a message which describes bilinear pairing groups, $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, with a bilinear pairing e from $\mathbb{G}_1, \mathbb{G}_2$ to $\mathbb{G}_T$, along with generators g_1, g_2 of groups $\mathbb{G}_1$ and $\mathbb{G}_2$, and q (as the order of the generators g_1, g_2). $\hat{\mathcal{N}}$ checks that g_1 and g_2 are indeed of order q, and $q \geq 2^k$. Otherwise, it and the functionality aborts.

It generates the CRS as in the real world, and sends the CRS to the adversary S' (we will reserve the name S for the simulator in the ideal world). It also saves the CRS as the state σ to be the starting state of each spawned $\mathcal{N}$.

On message (NewSession, *sid*, ssid, i, j, role) from adversary S', $\mathcal{N}$ (which gets pwd from the ideal functionality as auxiliary information) starts simulating a new instance of the protocol for P_i just as in the real world. That is it generates $R_1, S_2, T_1, \hat{\rho}_1$ just as in the real world using pwd, and random r_1 and s_1.

On NewKey call, it generates the key as in the real world as well, i.e. as

$$e(R'_2, \gamma_1) \cdot e((S'_2/\text{pwd}), \theta_1) \cdot e(T'_2, \rho_1) \cdot e(W_1, \hat{\rho}'_2),$$

where $R'_2, S'_2, T'_2, \hat{\rho_2}'$ were values generated by $\mathcal{N}$ in the peer session, and W_1 is the QA-NIZK as generated by the real world prover, i.e. $(\mathbf{w}_1 \cdot \mathbf{w}_2^{\iota_1})^{r_1}$.

On NewKeySim call, it gets a message $R'_2, S'_2, T'_2, \hat{\rho'}_2$ from S', and generates a message for S' as: Compute $\iota'_2 = \mathcal{H}(sid, \mathsf{ssid},\ P_j, P_i,\ R'_2, S'_2, \hat{\rho}'_2)$, if $(S'_2 \neq \text{pwd} \cdot (R'_2)^a)$ or $(T'_2 \neq (R'_2)^{d+\iota'_2 e})$ then send a random value in $\mathbb{G}_T$,
else, compute $\rho_1 = \mathbf{g}_2^{s_1}$, $\theta_1 = \mathbf{c}^{s_1}$, $\gamma_1 = (\mathbf{v}_1 \mathbf{v}_2^{\iota'_2})^{s_1}$, $W_1 = (\mathbf{w}_1 \mathbf{w}_2^{\iota_2})^{r_1}$, and send the message

$$e(R'_2, \gamma_1) \cdot e(S'_2/\text{pwd}, \theta_1) \cdot e(T'_2, \rho_1) \cdot e(W_1, \hat{\rho}'_2).$$

On corruption, it outputs ρ_1, θ_1 and γ_1 along with W_1 just as in the real world.

We now show that there is an interacting Turing machine $\hat{\mathcal{M}}$ such that it does not use pwd in generation of outgoing message for NewSession and for NewKey, and it generates the new session key randomly and independently. Indeed, $\hat{\mathcal{M}}$ behaves like simulator S in the proof of theorem 9.

It generates the CRS as the simulator S using these generators, and sends the CRS to the adversary S'. It also saves the CRS as the state σ to be the starting state of each spawned $\mathcal{N}$.

On message (NewSession, *sid*, ssid, i, j, role) from S', $\mathcal{N}$ starts simulating a new instance of the protocol for P_i just as S does. Thus, To simulate this instance, $\mathcal{N}$ chooses r_1, r'_1, r''_1, s_1 at random, and sets $R_1 = \mathbf{g}_1^{r_1}$, $S_1 = \mu \cdot \mathbf{a}^{r_1} \cdot \mathbf{g}_1^{r'_1}$, $\hat{\rho}_1 = \mathbf{b}^{s_1}$. Next, it computes $T_1 = \mathbf{g}_1^{r_1 \cdot (d_1 + \iota_1 e_1)} \cdot \mathbf{g}_1^{r''_1}$, where $\iota_1 = \mathcal{H}(sid, \mathsf{ssid},\ P_i, P_j,\ R_1, S_1, \hat{\rho}_1)$ (note the use of μ instead of pwd). It then sends $R_1, S_1, T_1, \hat{\rho}_1$ to S'. It retains r_1, r'_1, r''_1, s_1 (and μ if chosen randomly).

On receiving $(\mathsf{NewKey}, sid, \mathsf{ssid}, P_i)$ from S', it generates a random and independent key.

On NewKeySim call, it gets a message $R'_2, S'_2, T'_2, \hat{\rho'}_2$ from S', and generates a message for S' as: It computes pwd$'$ by decrypting S'_2, i.e. setting it to $S'_2/(R'_2)^a$. $\mathcal{S}$ then generates the following key for S':

$$\iota'_2 = \mathcal{H}(sid, \mathsf{ssid}, P_j, P_i, R'_2, S'_2, \hat{\rho}'_2),\ \rho_1 = \mathbf{g}_2^{s_1},\ \theta_1 = \mathbf{c}^{s_1},\ \gamma_1 = (\mathbf{v}_1\mathbf{v}_2^{\iota'_2})^{s_1},$$
$$W_1 = R_1^{(u'_1+\iota_1 u'_2)/b} \cdot (S_1/\mathrm{pwd}')^{c/b} \cdot T_1^{1/b}.$$

if $T'_2 = (R'_2)^{d+\iota'_2 e}$ then set $\xi_1 = \mathbf{1}_T$ else set $\xi = \epsilon(\mathbf{g}_1, \mathbf{g}_2)^{s'_1}$, where s'_1 is a fresh random value. Return the following key to S':

$$e(R'_2, \gamma_1) \cdot e((R'_2)^a, \theta_1) \cdot e(T'_2, \rho_1) \cdot e(W_1, \hat{\rho}'_2) \cdot \xi_1.$$

On corruption, it outputs ρ_1, θ_1 and γ_1, but W_1 simulated as

$$R_1^{(u'_1+\iota_1 u'_2)/b} \cdot (S_1/\mathrm{pwd})^{c/b} \cdot T_1^{1/b}$$

Recall, during the corruption call $\hat{\mathcal{M}}$ is allowed to use the auxiliary information pwd.

Lemma 27 *The view of an efficient and adaptive adversary S' interacting with $\hat{\mathcal{N}}$ and $\hat{\mathcal{M}}$ is computationally indistinguishable.*

The proof of this lemma is same as the proof of $\mathbf{Expt}_0$ and $\mathbf{Expt}_6$ in Section D. This proves that $\hat{\mathcal{N}}$ is a multi-session non-information oracle for password-based key exchange.

The proof of the theorem now starts with the UC simulator being at experiment $\mathbf{Expt}_6$ using $\hat{N}$. Rest of the proof is same as proof of theorem 9 from $\mathbf{Expt}_6 to \mathbf{Expt}_{11}$ which is the real world.

F Identity-Based Encryption

We demonstrate in this section that the concepts of DSS-QA-NIZK are ingrained in the fully secure IBE construction in [JR13], which in turn was inspired by the dual system IBE of [Wat09]. As in [JR13], the construction is not black-box but is better able to abstract the proof steps in the security proof. We start off by introducing a specialization of DSS-QA-NIZK with the concept of restriction-KeyGen.

DSS-QA-NIZK with Restriction KeyGen. We introduce the concept of a (PPT) *restriction-KeyGen* rK, which on input a label l and a CRS σ, generates a new (labeled-) CRS σ_l meant just for that label. The verifier is now required to be sound with respect to CRS σ_l, for all inputs with labels l. We also require similar restriction-KeyGen algorithms rpK and rsfK for the partial-simulation and one-time full-simulation worlds resp. The restriction-KeyGens rpK and rsfK also take a trapdoor as input.

The new formulation will continue to have PPT components such as all the CRS generation algorithms, P, V, as well as the proof simulators. But, in the security definitions, instead of the adversary having oracle access to the verifiers V, pV and sfV, it will have oracle access to the restriction-KeyGens rK, rpK and rsfK. In other words, the private verifiers are replaced by (private) restriction-KeyGens and the actual verifier in the simulation worlds remains the same as the real-world verifier V.

IBE Construction. We construct a DSS-QA-NIZK with restriction KeyGen for the following affine language L_ρ:

$$L_\rho = \{R, S \mid \exists r\ : R = \mathbf{g}_2^r, S = \mathbf{g}_2^u \cdot (\mathbf{g}_2^a)^r\}$$

where the parameter ρ is $(\mathbf{g}_2, \mathbf{g}_2^a, \mathbf{g}_2^u)$. To obtain a DSS-QA-NIZK, we attach an affine hash proof, i.e. the normal hash proof shifted by $\mathbf{u}$. This $\mathbf{u}$ is part of the prover CRS, so this affine hash proof can be generated easily from the normal hash proof. Following [JR13, JR14] one can give a QA-NIZK for affine languages in which the verifier CRS is independent of the affine constant, i.e., $\mathbf{u}$. Details of the construction are given in Appendix F.2. In addition to the standard DSS-QA-NIZK components, we additionally give restriction-KeyGen algorithms in the real, semi-functional and one-time full-simulation worlds. This construction is also *strongly split*.

We now give an IBE construction based on this strongly split DSS-QA-NIZK with restriction-KeyGens. However, the construction does not use the DSS-QA-NIZK as a black-box. Specifically, it uses a particular property of the construction for L_ρ: A labeled-CRS σ_l can be split as (σ', otp), such that the verifier $\mathsf{V}((\sigma', \mathsf{otp}), x, l, \pi)$, can be framed as checking $\mathsf{V}_L(\sigma', x, l, \pi) \overset{?}{=} \mathsf{otp}$, where $\mathsf{V}_L(\cdot,\cdot,\cdot,\cdot)$ is an efficient algorithm. The construction is given in Appendix F.3. In the subsequent sections, where we prove security of the scheme, we show that otp can be used as a one-time pad for encrypting the plaintext.

F.1 DSS-QA-NIZK with Restriction Keygen

We now consider a stronger version of DSS-QA-NIZK which is relevant for obtaining dual system IBE schemes. We introduce the concept of a *restriction keygen* which on input a "label" (and trapdoors) returns a new private verifier CRS which is meant for all proofs generated for that particular label.

In particular, a (PPT) *restriction-KeyGen* rK is required, which on input a label l and a CRS generates a new (labeled-) CRS CRSl meant just for that label. The verifier is now required to be sound with respect to CRS$_l$, for all inputs with labels l. We also require similar restriction-KeyGen algorithms rpK and rsfK for the partial-simulation and one-time full-simulation worlds resp. The restriction-KeyGens rpK and rsfK also take a trapdoor as input.

The new formulation will continue to have PPT components such as all the CRS generation algorithms, P, V, as well as the proof simulators. But, in the security definitions, instead of the adversary having oracle access to the verifiers V, pV and sfV, it will have oracle access to the restriction-KeyGens rK, rpK and rsfK. In other words, the private verifiers are replaced by (private) restriction-KeyGens and the actual verifier in the simulation worlds remains the same as the real-world verifier V.

Such a proof system is called a **DSS-QA-NIZK with restriction keygen** for a collection of witness relations $\mathcal{R}_\lambda = \{R_\rho\}$, with parameters sampled from a distribution $\mathcal{D}$, if for all non-uniform PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4)$ all of the following properties are satisfied:

- *restriction soundness:*
  $\Pr[\text{CRS} \leftarrow \mathsf{crsgen}(\lambda, \rho);\ (x, l, \pi) \leftarrow \mathcal{A}_0(\text{CRS}, \rho) :\ x \notin L_\rho\ \wedge\ \mathsf{ver}(\mathsf{rK}(\text{CRS}, l), x, l, \pi) = 1] \approx 0$

- *partial-ZK:*
  $\Pr[\lambda \leftarrow \mathsf{K}_0(1^m); \rho \leftarrow \mathcal{D}_\lambda;\ \psi\ \leftarrow\ \mathsf{K}_1(\lambda, \rho);\ \ \mathcal{A}_1^{\mathsf{P}(\psi,\cdot,\cdot,\cdot),\, \mathsf{rK}(\psi,\cdot)}(\psi) = 1]\ \approx$

$\Pr[\lambda \leftarrow \mathsf{K}_0(1^m); (\rho, \sigma, \tau, \eta) \leftarrow \mathsf{sfK}_1(\lambda);\ \mathcal{A}_1^{\mathsf{sfSim}^*(\sigma,\tau,\cdot,\cdot,\cdot),\,\mathsf{rpK}(\sigma,\eta,\cdot)}(\sigma) = 1]$,
where the calls to P (with inputs a witness and a language member) are restricted to ones satisfying R_ρ, **and** $\mathsf{sfSim}^*(\sigma, \tau, x, w, l)$ is defined to be sfSim $(\sigma,\ \tau,\ x,\ R_\rho(w, x),\ l)$ (i.e. witness is dropped, and membership bit β is just 1).

- *unbounded partial-simulation soundness:*
  $\Pr[\lambda \leftarrow \mathsf{K}_0(1^m); (\rho, \sigma, \tau, \eta) \leftarrow \mathsf{sfK}_1(\lambda);\ (x, l, \pi) \leftarrow \mathcal{A}_2^{\mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot),\,\mathsf{rpK}(\sigma,\eta,\cdot)}(\sigma) :$
  $\neg\exists w \text{ s.t. } R_\rho(w, x) = 1, \text{ and } \mathsf{V}(\mathsf{rpK}(\sigma, \eta, l), x, \pi) = 1] \approx 0.$

- *one-time full-ZK:*
  $\Pr[\lambda \leftarrow \mathsf{K}_0(1^m); (\rho, \sigma, \tau, \eta) \leftarrow \mathsf{sfK}_1(\lambda);\ (x, l, \beta, s) \leftarrow \mathcal{A}_3^{\mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot),\,\mathsf{rpK}(\sigma,\eta,\cdot)}(\sigma);$
  $\pi \leftarrow \mathsf{sfSim}(\sigma, \tau, x, \beta; l) : \mathcal{A}_4^{\mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot),\,\mathsf{rpK}(\sigma,\eta,\cdot)}(\pi, s) = 1] \ \approx$
  $\Pr[\lambda \leftarrow \mathsf{K}_0(1^m); \rho \leftarrow \mathcal{D}_\lambda;\ (\sigma, \tau, \tau_1, \eta)\ \leftarrow\ \mathsf{otfK}_1(\lambda, \rho);$
  $(x, l, \beta, s) \leftarrow \mathcal{A}_3^{\mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot),\,\mathsf{rsfK}(\sigma,\eta,\cdot)}(\sigma);$
  $\pi \leftarrow \mathsf{otfSim}(\sigma, \tau_1, x, l) : \mathcal{A}_4^{\mathsf{sfSim}(\sigma,\tau,\cdot,\cdot,\cdot),\,\mathsf{rsfK}(\sigma,\eta,\cdot)}(\pi, s) = 1]$,
  where β is a correct L_ρ-membership bit for x, **and** all calls to sfSim also have correct L_ρ-membership bits **and** label l is not invoked on $\mathsf{rsfK}/\mathsf{rpK}$. Here s is a state variable.

A DSS-QA-NIZK with restriction keygen is called a **strongly split** if in the partial simulation world (1) the verifier CRS $\mathbf{CRS}_v$ output by the semi-functional CRS simulator sfK_1 is independent of the language parameter ρ that sfK_1 chooses, (2) the proof simulator trapdoor τ is also independent of this language parameter ρ, and (3) the semi-functional simulator sfSim does not use $\mathbf{CRS}_p$ (i.e. only needs trapdoor τ).

In addition, for facilitating IBE constructions, A labeled-CRS σ_l can be split as (ψ', otp), such that the verifier $\mathsf{V}((\psi', \mathsf{otp}), x, l, \pi)$, can be framed as checking $\mathsf{V}_L(\psi', x, l, \pi) \overset{?}{=} \mathsf{otp}$, where $\mathsf{V}_L(\cdot, \cdot, \cdot, \cdot)$ is an efficient algorithm.

F.2 Construction for an Affine DDH Language

We construct a DSS-QA-NIZK with restriction keygen for the following affine language L_ρ:

$$L_\rho = \{R, S \mid \exists r\ : R = \mathbf{g}_2^r, S = \mathbf{g}_2^u \cdot (\mathbf{g}_2^a)^r\}$$

where the parameter ρ is $(\mathbf{g}_2, \mathbf{g}_2^a, \mathbf{g}_2^u)$. This construction will also be *strongly split.*

We now describe the various components.

- The real world algorithms:
  - The algorithm K_0 is just the group generation algorithm (it takes a unary string 1^m as input). The CRS generation algorithm K_1 takes language parameter ρ (as above) and the group parameters as input and generates the CRS as follows: it chooses $b, d_1, d_2, e_1, e_2, \Delta_1', \Delta_2', \Delta_3', \Delta_4'$ at random from $\mathbb{Z}_q$ and computes $\mathbf{w}_1 = \mathbf{g}_2^{(\Delta_1'+d_1+ad_2)/b}$, $\mathbf{w}_2 = \mathbf{g}_2^{(\Delta_2'+e_1+ae_2)/b}$, $\mathbf{w}_3 = \mathbf{g}_2^{(\Delta_3'+a)/b}$, $\mathbf{w}_4 = \mathbf{g}_2^{(\Delta_4'+u)/b}$, $\mathbf{d} = \mathbf{g}_2^{d_1+ad_2}$, $\mathbf{e} = \mathbf{g}_2^{e_1+ae_2}$, and publishes the prover CRS as

$$\mathbf{CRS}_p = \{\mathbf{g}_2, \mathbf{g}_2^a, \mathbf{g}_2^u, \mathbf{d}, \mathbf{e}, \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \mathbf{w}_4\}.$$

As for the verifier CRS it computes $\mathbf{v}_1 = \mathbf{g}_1^{-\Delta'_1}$, $\mathbf{v}_2 = \mathbf{g}_1^{-\Delta'_2}$, $\mathbf{v}_3 = \mathbf{g}_1^{-\Delta'_3}$, $\mathbf{k} = e(\mathbf{g}_1, \mathbf{g}_2)^{\Delta'_4}$, and publishes

$$\mathsf{CRS}_v = \{\mathbf{g}_1, \mathbf{g}_1^b, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{k}\}.$$

– The **prover** P takes as input CRS_p, a language member $\langle R, S\rangle$ and its witness r and a label i and produces a proof π consisting of three group $\mathbb{G}_2$ elements T, W_1 and W_2 as follows: Compute $T = (\mathbf{d}\mathbf{e}^{i})^r$, $W_1 = (\mathbf{w}_1 \cdot \mathbf{w}_2^{i})^r$, and $W_2 = \mathbf{w}_4 \cdot \mathbf{w}_3^r$.

– The **restriction keygen** rK takes as input the verifier CRS and a label i and first picks a random value s from $\mathbb{Z}_q$, and a TAG from Z_q and returns the CRS

$$\{\mathbf{g}_1^s, \mathbf{g}_1^{bs}, (\mathbf{v}_1\mathbf{v}_2^{i}\mathbf{v}_3^{\text{TAG}})^s, \text{TAG}, \mathbf{k}^s\}.$$

– The **verifier** V takes the CRS of the syntax returned by the restriction verifier, say $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \text{TAG}, \mathsf{otp}$, and a potential language member R, S (i.e. no label or ignores label) and checks if

$$\left(e(\mathbf{z}_3, R) \cdot e(\mathbf{z}_1^{-1}, S^{\text{TAG}} \cdot T) \cdot e(\mathbf{z}_2, W_1 \cdot W_2^{\text{TAG}})\right)^{1/\text{TAG}} \stackrel{?}{=} \mathsf{otp}$$

holds.

- The partial-simulation world PPT components:

– The **semi-functional CRS simulator** sfK_1 takes group parameters as input and produces a witness relation parameter ρ as $(\mathbf{g}_2, \mathbf{g}_2^a, \mathbf{g}_2^u)$, by picking a, u at random from $\mathbb{Z}_q$. Next, it produces the semi-functional CRS $\sigma = (\mathsf{CRS}_p, \mathsf{CRS}_v)$ exactly as in the real world. Let $d = d_1 + ad_2$ and $e = e_1 + ae_2$. It outputs $b, d, e, \Delta'_1, \Delta'_2, \Delta'_3, \Delta'_4$ as proof simulator trapdoors τ, and outputs a, u, d, e as private-verifier trapdoors η.

– The **semi-functional simulator** sfSim ignores prover CRS and uses trapdoors $\{b, d, e, \Delta'_1, \Delta'_2, \Delta'_3, \Delta'_4\}$ to produce a proof for a potential language member $\langle R, S\rangle$ and a label i, and a membership bit β as follows: if $\beta = 0$ let $T = \mathbf{g}_1^y$, where y is a new random value from $\mathbb{Z}_q$ else let $T = R^{d+ie}$, and then compute:

$$W_1 = R^{(\Delta'_1 + i\Delta'_2)/b} \cdot T^{1/b},\ W_2 = \mathbf{g}_1^{\Delta'_4/b} \cdot R^{\Delta'_3/b} \cdot S^{1/b}.$$

– The **restriction private KeyGen** rpK uses a verifier CRS and trapdoors $\{a, u, d, e\}$, and on input i, outputs the following verifier-CRS : it first picks $s.s'$, TAG randomly and independently from $\mathbb{Z}_q$, and outputs the restriction verifier-CRS:

$$\mathbf{g}_1^{s+s'},\ \mathbf{b}^s,\ (\mathbf{v}_1\mathbf{v}_2^{i}\mathbf{v}_3^{\text{TAG}})^s \cdot \mathbf{g}_1^{(d+ie+\text{TAG}\cdot a)\cdot s'},\ \text{TAG},\ \mathbf{k}^s \cdot e(\mathbf{g}_1,\ \mathbf{g}_2)^{u\cdot s'}.$$

- Finally, the one-time full simulation PPT components:

– The **one-time full-simulation CRS simulator** otfK_1 takes group parameters and the language parameters ρ (as above) as input and produces the CRS $\sigma = (\mathsf{CRS}_p, \mathsf{CRS}_v)$ exactly as in the real world It outputs $\mathbf{g}_2^u, b, d_1, d_2, e_1, e_2, \Delta'_1, \Delta'_2, \Delta'_3, \Delta'_4$ as all three trapdoors τ, τ_1, η.

– The **one-time full simulator** otfSim uses trapdoors $\{\mathbf{g}_2^u, b, d_1, d_2,\ e_1, e_2,\ \Delta'_1,\ \Delta'_2,\ \Delta'_3,\ \Delta'_4\}$ to produce a proof for a potential language member $\langle R, S\rangle$ and label i as follows:

$$T = R^{d_1 + ie_1} \cdot (S/\mathbf{g}_2^u)^{d_2 + ie_2},\ W_1 = R^{(\Delta'_1 + i\Delta'_2)/b} \cdot T^{1/b},\ W_2 = \mathbf{g}_1^{\Delta'_4/b} \cdot R^{\Delta'_3/b} \cdot S^{1/b}.$$

– The **restriction semi-functional keygen** rsfK uses a verifier CRS and trapdoors $\{\mathbf{g}_2^u,\ b,\ d_1, d_2, e_1, e_2,\ \Delta'_1, \Delta'_2.\Delta'_3, \Delta'_4\}$, and on input i, outputs the following verifier-CRS : it first picks $s.s'$ randomly and independently from $\mathbb{Z}_q$, sets $\text{TAG} = -(d_2 + ie_2)$, and outputs the restriction verifier-CRS

$$\mathbf{g}_1^{s+s'},\ \mathbf{g}_1^{bs},\ (\mathbf{v}_1\mathbf{v}_2^i\mathbf{v}_3^{\text{TAG}})^s \cdot \mathbf{g}_1^{(d_1+ie_1)\cdot s'},\ \text{TAG},\ \mathbf{k}^s \cdot e(\mathbf{g}_1, \mathbf{g}_2)^{u\cdot s'}.$$

F.3 IBE Construction Using DSS-QA-NIZK

We now give an IBE construction based on this strongly split DSS-QA-NIZK with restriction-KeyGens.

- **Setup:** The trusted authority chooses language parameters $\rho = \{\mathbf{g}_2, \mathbf{g}_2^a, \mathbf{g}_2^u\}$ by choosing the bilinear group parameters according to a group generation algorithm for which the SXDH assumption holds, and then picking a and u randomly and independently from $\mathbb{Z}_q$. Next it generates CRS_p and CRS_v using the CRS generation algorithm K_1. It publishes the **public key** as CRS_v. It retains ρ and the CRS_p as the **master secret key**.

- **KeyGen:** For an identity i, the private key for that identity is generated using master secret key as follows: Essentially, generate a fresh random pair $\langle R, S\rangle$ from the language L_ρ. Next generate a proof π with label i using the prover P and CRS_p. The private key for identity i is output as R, S, π.

- **Encrypt:** To encrypt message $M \in \mathbb{G}_T$, for identity i, call restriction-KeyGen rK with input i to get a labeled-CRS, say (ψ'', otp). The ciphertext is: $(M \cdot \mathsf{otp}, \psi'')$

- **Decrypt:** Given a ciphertext $\langle C_0, C_1\rangle$ purportedly for identity i, decrypt using the private key for i as follows: Use the verifier component algorithm V_L on the ciphertext viewed as a labeled-CRS, and output $C_0/\mathsf{V}_L(C_1, \langle R, S\rangle, i, \pi)$.

F.4 Proof of security of IBE Construction

We show that otp (as used in blinding the plaintext M) is distributed randomly in the view of an adaptive Adversary, who after obtaining the public key, adaptively obtains private keys for multiple identities $i_1, i_2, ..., i_n$, and a ciphertext for identity i (where all the identities are chosen adaptively by the Adversary, and i is different from the secret key identities). The ciphertext can be obtained by the Adversary at any stage. We prove this below, by going through a sequence of games, starting by moving to the partial-simulation world. Moving to this world enables us to replace, one by one, the private keys using a valid language member (with proof generated by calling sfSim with $\beta = 1$) to ones using a fake language member (with proof generated by calling sfSim with $\beta = 0$), by intermediately transitioning to the one-time full simulation world.

Proof: We will just show that otp (as used in blinding the plaintext M) is distributed randomly in the view of an adaptive Adversary, who after obtaining the public key, adaptively obtains private keys for multiple identities $i_1, i_2, ..., i_n$, and a ciphertext for identity i (where all the identities are chosen adaptively by the Adversary, and i is different from the secret key identities). The ciphertext can be obtained by the Adversary at any stage.

We will consider a sequence of games, and show that the Adversary's view is either statistically or computationally indistinguishable between any two consecutive games. Game G_0 is same as the actual adaptive security IBE game of the previous paragraph.

Game $\mathbf{G}_1$: In this game we emulate the partial-simulation world, but with all private keys generated properly. More precisely, the setup is done using semi-functional CRS simulator sfK_1, and the challenger retains the simulation trapdoor τ and the private-verifier trapdoor η. The *KeyGen* on input an identity i is done using semi-functional simulator sfSim and trapdoor τ (note, since our construction is strongly split, CRS_p is not required for this simulated proof generation). The membership bit β is 1 in all such calls to sfSim, as indeed each of the pairs R, S is honestly generated. The message is encrypted (for identity i) using the restriction-CRS returned by rpK (which used trapdoors η and CRS_v). The decryption is as before using rV.

By partial-ZK property of the DSS-QA-NIZK, the Adversary's view in Games $\mathbf{G}_0$ and $\mathbf{G}_1$ is computationally indistinguishable.

Game $\mathbf{G}_2$: This game is identical to $\mathbf{G}_1$ except that the pairs R, S in each KeyGen is generated differently. More precisely, for the j-th KeyGen, say for identity i_j, the R, S are chosen as follows: choose r_j and r'_j randomly and independently from $\mathbb{Z}_q$, and let $R = \mathbf{g}_2^{r_j}$ and $S = \mathbf{g}_2^{r'_j}$. Of course, the proof simulator sfSim is now invoked with membership bit β set to zero.

We now prove that the view of the Adversary in games $\mathbf{G}_1$ and $\mathbf{G}_2$ is computationally indistinguishable. This is proven by induction over j via several hybrid games where in each subsequent hybrid game (say j-th hybrid game) we alter the generation of R, S in the j-th KeyGen from honest (i.e. as in game $\mathbf{G}_1$) to fake as described in game $\mathbf{G}_2$. Computational indistinguishability of two such consecutive hybrid games is shown by first switching to the one-time full simulation world and generating proof in the j-th KeyGen using the one-time full simulator otfSim. This one-time full -simulation allows us to employ DDH to replace an honest S (i.e. $\mathbf{g}^u \cdot \mathbf{g}_2^{ar_j}$) by $\mathbf{g}_2^u \cdot \mathbf{g}_2^{ar'_j}$. Next, we switch back to the partial simulation world (and this time supply $\beta = 0$ to the proof simulator sfSim in the j-th KeyGen). S can also be just made $\mathbf{g}_2^{r'_j}$, as r'_j is not used anywhere else.

Now, lets analyze the view of the Adversary in game $\mathbf{G}_2$. The public key CRS_v is independent of u since by the strongly split property (1) of the DSS-QA-NIZK CRS_v is independent of ρ. Next, the private keys for all the identities sought by the Adversary are independent of u: each R, S for the j-th key is independent of u, and further the proof generated on such R, S uses trapdoor τ and does not use CRS_p (again by the strongly split property (3)), and τ itself is independent of ρ by strongly split property (2). Thus, the only elements which the Adversary gets which could depend on u maybe from the ciphertext (i.e. the verifier CRS output by rpK). Indeed, a simple examination shows that $\mathsf{otp} = \mathbf{k}^s \cdot e(\mathbf{g}_1, \mathbf{g}_2)^{u \cdot s'}$, and all the other elements in the verifier CRS returned by rpK are independent of u. Since otp was used as the one-time pad in the encryption of M, it follows that this one-time pad is random and independent of the Adversary's view (because u is chosen randomly, and s' is non-zero with high probability). That completes the proof. $\square$ $\square$