Round-Optimal Blind Signatures in the Plain Model from Classical and Quantum Standard Assumptions

Shuichi Katsumata[1], Ryo Nishimaki[2], Shota Yamada[1], Takashi Yamakawa[2]

[1]National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
{shuichi.katsumata,yamada-shota}@aist.go.jp
[2]NTT Secure Platform Laboratories, Tokyo, Japan
{ryo.nishimaki.zk,takashi.yamakawa.ga}@hco.ntt.co.jp

March 8, 2021

Abstract

Blind signatures, introduced by Chaum (Crypto'82), allows a user to obtain a signature on a message without revealing the message itself to the signer. Thus far, all existing constructions of round-optimal blind signatures are known to require one of the following: a trusted setup, an interactive assumption, or complexity leveraging. This state-of-the-affair is somewhat justified by the few known impossibility results on constructions of round-optimal blind signatures in the plain model (i.e., without trusted setup) from standard assumptions. However, since all of these impossibility results only hold *under some conditions*, fully (dis)proving the existence of such round-optimal blind signatures has remained open.

In this work, we provide an affirmative answer to this problem and construct the first round-optimal blind signature scheme in the plain model from standard polynomial-time assumptions. Our construction is based on various standard cryptographic primitives and also on new primitives that we introduce in this work, all of which are instantiable from *classical and post-quantum* standard polynomial-time assumptions. The main building block of our scheme is a new primitive called a blind-signature-conforming zero-knowledge (ZK) argument system. The distinguishing feature is that the ZK property holds by using a quantum polynomial-time simulator against non-uniform classical polynomial-time adversaries. Syntactically one can view this as a delayed-input three-move ZK argument with a reusable first message, and we believe it would be of independent interest.

1 Introduction

1.1 Background

Blind signatures enable users to obtain a signature without revealing a message to be signed to a signer. More precisely, a blind signature scheme is a two-party computation between a signer and a user. The signer has a pair of keys called verification-key and signing-key, and the user takes as input a message and the verification-key. They interact with each other, and the user obtains a signature for the message after the interaction. There are two security requirements on blind signatures: (1) users cannot forge a signature for a new message (unforgeability), and (2) the signer cannot obtain information about the signed messages (blindness).

Chaum introduced the notion of blind signatures and provided a concrete instantiation, while also showing an application to e-cash systems [Cha82]. After its invention, blind signatures have been used as a crucial building block for various other privacy-preserving crypto-systems such as e-voting [FOO93, Cha88], anonymous credential [CL01], and direct anonymous attestation [BCC04].

Round-complexity. One of the main performance measures for blind signatures is round-complexity. A round-optimal blind signature is a blind signature with only 2-moves[1], where the user and signer sends one message to each other. We focus on round-optimal blind signatures in this study since a high round-complexity is one of the main bottlenecks in cryptographic systems. Another advantage is that round-optimal blind signatures are automatically secure in the concurrent setting [Lin08, HKKL07].

Round-optimal scheme in the plain model from standard assumptions. From a theoretical point of view, using less and weaker assumptions is much better. However, all existing round-optimal blind signature schemes require either (1) a trusted setup [Fis06, AO12, AFG+16, BFPV11, BPV12, MSF10, SC12, Bol03, BNPS02], (2) an interactive assumption [FHS15, FHKS16, Gha17, BNPS02, Bol03], or (3) complexity leveraging [GRS+11, GG14]. We briefly discuss each item. In the trusted setup model, if an authority set a backdoor, we can no longer guarantee any security. Interactive assumptions are non-standard compared to standard non-interactive ones since an adversary can interact with the challenger.[2] Complexity leveraging uses a gap between the computational power of an adversary and the reduction algorithm in security proofs. To create this gap, we require super-polynomial-time assumptions[3] and large parameters, which hurt the overall efficiency. In fact, there are a few impossibility results on constructing round-optimal blind signatures in the plain model (i.e., without any trusted setup) from standard assumptions *under some conditions* [Lin08, FS10, Pas11]. So far, constructing a round-optimal blind signature scheme in the plain model from standard polynomial-time assumptions has proven to be elusive.

Thus, a natural and long-standing open question is the following:

Can we achieve a round-optimal blind signature scheme in the plain model from standard polynomial-time assumptions?

We affirmatively answer this open question in this study. Hereafter, we call blind signatures that satisfy all the above conditions as a blind signature with desired properties.

1.2 Our Result

We present a round-optimal blind signature scheme with desired properties. Our construction relies on various standard cryptographic primitives such as oblivious transfer and also new primitives that we introduce in this study, all of which are instantiable from *classical and quantum* standard polynomial-time assumptions.[4]

Our construction is based on the idea by Kalai and Khurana [KK19] that we can replace complexity leveraging with classical and quantum assumptions. However, our technique is not a simple application of their idea. There are several technical hurdles to avoid complexity leveraging in blind signatures, even if we use classical and quantum assumptions. We provide further details in Section 1.3.

The main building block of our scheme is a blind-signature-conforming zero-knowledge (ZK) argument system, which we introduce in this study. It is a 2-move ZK argument system in the reusable public key model where the ZK property holds by using a quantum polynomial-time simulator against non-uniform classical polynomial-time adversaries, and parties have access to a reusable public key (possibly maliciously) generated by a prover. We construct a blind-signature-conforming ZK argument for any NP language from standard classical and quantum assumptions. We give an overview of our technique in Section 1.3.

Although our scheme satisfies desirable features in the theoretical sense, it is not quite practical since we rely on general cryptographic tools such as garbled circuits. We believe our scheme opens the possibility of practical round-optimal blind signatures with the desired properties. We leave this question as an open problem.

1.3 Technical Overview

Here, we provide an overview of our construction.

[1] We count one move when an entity sends information to the other entity.

[2] An adversary may have the flexibility to choose a problem instance or obtain auxiliary information related to a problem instance.

[3] A super-polynomial-time assumption means that a hard problem cannot be broken even by *super-polynomial-time* adversaries. This is stronger than a standard polynomial-time assumption, where adversaries are restricted to run in polynomial-time.

[4] The learning with errors (LWE) assumption against quantum polynomial time adversaries and one of the following assumptions against (non-uniform) classical polynomial time adversaries: quadratic residuosity (QR), decisional composite residuosity (DCR), symmetric external Diffie-Hellman (SXDH) over pairing group, or decisional linear (DLIN) over pairing groups.

Blind signature scheme by Garg et al. Our starting point is the blind signature scheme by Garg, Rao, Sahai, Schröder, and Unruh [GRS+11]. Their scheme is round-optimal and in the plain model, but the security proof requires complexity leveraging. Our goal is to remove the complexity leveraging and base the security on classical and quantum polynomial assumptions.

Here, we recall their construction. In their protocol, a signer publishes a verification key of a digital signature scheme as its public key and keeps the corresponding signing key secret. To blindly sign on a message, the signer and the user run secure function evaluation (SFE) protocol where the signer plays the role of the sender and the user plays the role of the receiver. In more detail, the user's input is the message to be signed, and the signer holds a circuit corresponding to the signing algorithm of the digital signature scheme where the signing key is hardwired. At the end of the protocol, only the user receives the output signature. To prevent malicious behaviors of the signer, such as using arbitrarily chosen randomness for the signing algorithm to break the blindness, they make the signing algorithm deterministic by using a PRF and include the perfectly binding commitment of the signing key into the public key. Furthermore, they have the signer prove that it honestly follows the SFE protocol using a zero-knowledge argument system.

The blindness of the protocol follows from the receiver's security of the SFE and from the fact that the signer cannot deviate from the honest execution of the protocol due to the soundness of the zero-knowledge argument system and the binding property of the commitment scheme. On the other hand, the unforgeability follows from the combination of the zero-knowledge property of the zero-knowledge argument system, the sender's security of the SFE protocol, and the unforgeability of the digital signature scheme. The former two properties intuitively imply that the user cannot obtain anything beyond the signatures corresponding to the messages it chooses. The final property implies that it cannot forge a new signature. While intuitively correct, there are two problems with this approach. The first problem is with the reduction algorithm that reduces the unforgeability of the blind signature scheme to that of the underlying digital signature scheme. The reduction algorithm has to simulate the signer and extract the message to be signed from the first message of the user. However, this should not be possible because of the receiver's security of the SFE. The second problem is that we need a 2-move zero-knowledge argument system to obtain round-optimal blind signatures. However, it is known that a 2-move zero-knowledge argument system is impossible [GO94].

To resolve these problems, they assume super-polynomial security for the underlying (plain) signature scheme and allow the corresponding reduction algorithm to run in super-polynomial time. Then, the first issue can be resolved by letting the reduction algorithm *break* the receiver's security of the SFE scheme and extract the message to be signed using its super polynomial power. Furthermore, allowing the reduction algorithm to run in super-polynomial time also enables them to sidestep the impossibility result mentioned above. They use a 2-move zero-knowledge argument system with a super-polynomial time simulator by Pass [Pas03] and run the super-polynomial time simulator in the reduction algorithm for unforgeability.[5] This also resolves the second issue above.

Our first step towards the goal is to replace the super-polynomial time reduction algorithm in their security proof with a quantum-polynomial time (QPT) algorithm, which is inspired by Kalai and Khurana [KK19]. To make this work, we replace primitives with super-polynomial security with quantumly secure ones and the primitives broken by the super-polynomial time algorithm with quantumly insecure and classically secure ones. However, simple replacement of the underlying primitives does not work, because their security proof uses complexity leveraging twice, which requires three levels of security for the underlying primitives, while the combination of classical and quantum polynomial hardness can offer only two levels of security.[6] In particular, the above idea necessitates 2-move zero-knowledge arguments with QPT simulation, which cannot be obtained by a simple modification of the construction by Pass [Pas03]. As we elaborate in the following, we relax the notion of zero-knowledge argument system so that it still implies blind signatures and provides a construction that satisfies the notion by adding many modifications to the original zero-knowledge argument system by Pass [Pas03].

Zero-Knowledge argument system by Pass. To see the problem more closely, we review the zero-knowledge argument system by Pass [Pas03], which is used in the construction of round-optimal blind signatures by Garg et al. [GRS+11]. Their starting point is ZAP for NP languages [DN00, DN07]. Recall that ZAP is a 2-move public coin

[5] Though Garg et al. [GRS+11] does not explicitly state that they use the zero-knowledge argument of [Pas03], we observe that their construction can be viewed in this way.

[6] A reader might consider starting from the blind signature scheme by Garg and Gupta [GG14] instead since their security proof uses complexity leveraging only once. However, their construction may not be compatible with our idea of using quantum simulation since it is heavily dependent on a specific structure of the Groth-Sahai proofs [GS08], which is quantumly insecure.

witness indistinguishable proof system without setup, where the first message can be reused. To make it zero-knowledge, they use the "OR-proof trick" by Feige, Lapidot, and Shamir [FLS90, FLS99]. This technique converts a witness indistinguishable proof into a zero-knowledge proof in the context of non-interactive proof systems by adding a trapdoor branch for the relation to be proven so that the zero-knowledge simulator can use the branch. In more detail, the protocol proceeds as follows.

1. In the first round of the protocol, the verifier sends the first round message r_{zap} of the ZAP system along with a random image $z = f(y)$ of a one-way permutation (OWP) $f : \{0,1\}^\ell \to \{0,1\}^\ell$.[7]

2. Given the message, the prover who proves $x \in \mathcal{L}$, where $\mathcal{L}$ is some NP language specified by a relation R, proceeds as follows. It first commits the string 0^ℓ by a non-interactive commitment with perfect biding to obtain $\mathsf{com} = \mathsf{Com}(0^\ell; r_{\mathsf{com}})$ using randomness r_{com}. It then proves that there is witness $(w', y', r'_{\mathsf{com}})$ such that

$$\Big((x, w') \in R\Big) \vee \Big(\mathsf{com} = \mathsf{Com}(y'; r'_{\mathsf{com}}) \wedge f(y') = z\Big)$$

   by the proving algorithm of the ZAP system to obtain a proof π_{zap} and sends $\pi = (\mathsf{com}, \pi_{\mathsf{zap}})$ to the verifier. Note that in the honest execution, the prover sets $(w', y', r'_{\mathsf{com}}) = (w, \bot, \bot)$.

3. Given the proof from the prover, the verifier parses $\pi \to (\mathsf{com}, \pi_{\mathsf{zap}})$ and verifies the proof π_{zap} for the above statement by the verification algorithm of the ZAP system.

We then discuss the security of the system. Let us start with the zero-knowledge property. As mentioned, the simulator will run in super-polynomial time, say T. Given (r_{zap}, z), the simulator uses its super-polynomial power to invert the permutation to compute $y = f^{-1}(z)$. It then computes a commitment $\mathsf{com} = \mathsf{Com}(y; r_{\mathsf{com}})$ and uses the witness $(w', y', r'_{\mathsf{com}}) = (\bot, y, r_{\mathsf{com}})$ to generate a proof. Due to the witness indistinguishability of the underlying ZAP and the hiding property of the commitment, the simulated proof is indistinguishable from the real one. The proof for soundness is a bit more complicated. Let us assume an adversary that can generate an accepting proof for a false statement $x \notin \mathcal{L}$. By the statistical soundness of ZAP and by the fact that $x \notin \mathcal{L}$, the output (com^*, π^*) of the successful adversary should satisfy the trapdoor branch of the relation. Namely, com^* should be a commitment of $y = f^{-1}(z)$. Intuitively, this contradicts the one-wayness of f, and thus the system is sound since generating such a commitment seems to require the knowledge of y. However, to turn this intuition into a formal argument, we have to construct a reduction algorithm (i.e., inverter for the OWP) that outputs $y = f^{-1}(z)$ in the clear, instead of the commitment of y. To do so, they turn to complexity leveraging. Namely, they consider an inversion algorithm that runs in super-polynomial time, say T', and have the algorithm extract y from com^* using its super-polynomial power. If we assume f is hard to invert in time T' and the commitment is broken in time T', we can derive the contradiction as desired.

We observe that the two super-polynomial functions T and T' should satisfy $T \gg T'$, since f should be invertible in time T for the zero-knowledge simulator to work, while f should be hard to invert in time T' for the above reduction to make sense. This seems to be incompatible with our approach of replacing T-time simulator with QPT simulator, since this requires hardness that lies between QPT hardness and classical polynomial hardness to replace T'-time secure primitives with something. However, we do not know how to do this without turning to complexity leveraging.

Replacing the commitment with encryption. As we observed above, the main technical hurdle to our goal is that there is no efficient way to extract the message from the commitment for the reduction algorithm that inverts the OWP. However, extraction should not be possible efficiently, since otherwise the commitment cannot be hiding and thus harms the zero-knowledge property. To satisfy these contradicting requirements, we switch to the non-uniform setting and use the standard trick of leveraging the gap between the information available for algorithms in the real-world and non-uniform reduction algorithms. As observed by Garg et al. [GRS+11], non-uniform algorithms can be regarded as two-stage algorithms. The pre-computation phase of the algorithm takes the security parameter as input and computes an advice string of polynomial length using *unbounded computational power*. Then, the online phase of the algorithm takes the problem instance along with the advice string as input and tries to solve the problem in polynomial time. In our context, the non-uniform reduction algorithm will use this advice string to efficiently extract the message from the

[7] Though one-way functions with efficiently decidable images suffice, we use OWP in this overview for simplicity. In our construction, we rely on a slightly generalized notion of *hard problem generators* which we introduce in Section 3.1.

commitment. On the other hand, this advice string is not available for the real world algorithms and hence does not harm the hiding property of the commitment.

To implement this idea, we replace the commitment with public key encryption (PKE) and change the protocol so that the prover encrypts 0^ℓ using a public key pk_P chosen by itself, instead of computing a commitment of 0^ℓ. The advice string in our context is the secret key corresponding to pk_P. Using the secret key, one can efficiently decrypt the ciphertext and extract the message as desired. Subtle yet, the important point is that the prover should choose the public key pk_P *before* the protocol is run and use the same public key for every invocation of the protocol. Then, the non-uniform reduction algorithm can find the secret key corresponding to pk_P in the pre-computation phase using its unbounded computational power, since pk_P is chosen before the problem instance $z = f(y)$ of the OWP is chosen. This is not possible if the prover chooses a fresh public key for every encryption because the problem instance z and the public key are chosen at the same time in this case. It is not possible to off-load the task of finding the secret key to the pre-computation phase.

In fact, with the above modification, the argument system is no longer in the plain model, since we allow the prover to choose a long-term public key. However, since the syntax of round optimal blind signatures allows the signer to have a long-term public parameter, this modification does not affect the application to blind signatures.

Dealing with maliciously generated public keys. While the above idea may seem to work at first sight, there is still an issue. The problem is that a malicious prover may choose an ill-formed public key for the PKE, for which there are no corresponding secret keys. In this case, we may not be able to extract the message from the ciphertext even with unbounded computational power. We should consider this kind of attack since a malicious signer against blind signatures may maliciously choose a public key. A simple countermeasure against this attack would be to use a PKE scheme such that one can efficiently decide whether the public key is honestly generated or not and have the verifier reject provers with ill-formed public keys. However, we cannot adopt this simple solution because we do not know how to instantiate such a PKE. In particular, we require the PKE to have security against QPT adversaries in addition to the above property due to a technical reason,[8] but there are no known PKE schemes satisfying these properties simultaneously.

To resolve the issue, we further change the protocol. Our first attempt is to let the verifier choose a public key pk_V of PKE and have the prover encrypt 0^ℓ under pk_V in addition to the long-term public key pk_P. Furthermore, we have the prover prove that it has valid witness w for x or it encrypts y under pk_P and pk_V. With this change, the reduction algorithm can extract the message from the ciphertext corresponding to pk_V even if pk_P is maliciously generated since pk_V is under the control of the reduction algorithm and honestly generated. However, this modification harms the zero-knowledge property. In particular, since the verifier has secret key corresponding to pk_V, it can know whether the proof is generated from the honest execution of the protocol or not by simply decrypting the ciphertext.

The reason why the above idea fails is that we allow too much flexibility for the verifier in the sense that it can choose a public key that enables the extraction even if the prover behaves honestly. What we really need is a mechanism where the verifier can extract the message only when the prover cheats. For this purpose, we use lossy encryption. Recall that lossy encryption [PVW08, BHY09] is an extension of PKE where we have an additional lossy key generation algorithm. While the normal key generation algorithm outputs a public key and secret key, the lossy key generation algorithm only outputs a public key. For lossy encryption, we require the lossiness property, which stipulates that the ciphertext generated under the lossy key does not carry any information of the message. As for security, we require that the lossy key and the normal key are indistinguishable. We then would like to change the protocol so that the verifier is restricted to choose the lossy public key in the honest execution of the protocol and can choose normal public key that allows the extraction only when the prover chooses an ill-formed public key. To restrict the behavior of the verifier, we have the verifier prove the following statement:

$$(\mathsf{pk}_V \text{ is chosen from the lossy key generation}) \vee (\mathsf{pk}_P \text{ is an ill-formed public key})\,. \tag{1}$$

The former branch of the statement is used in the honest execution and the latter is for simulation. The proof is generated by running another instance of the ZAP system, where the roles of the prover and the verifier are swapped. To avoid

[8] We need security against QPT adversaries for the PKE scheme because its security is used to prove zero-knowledge property, where the simulator is a QPT algorithm. Recall that the simulator needs quantum power to invert the OWP. One may try to show that non-uniform security instead of quantum security is enough for the PKE by using the pre-computation trick we mentioned. However, this does not seem possible because the inversion should be done *after* the public key is chosen.

increasing the round of the overall protocol, we put the first round message of the ZAP system into the public parameter of the prover and have the verifier generate the proof with respect to it and send the proof along with pk_V to the prover in the first round. Note that it is not clear how to prove the above statement by the ZAP system, since it is not necessarily in NP. In particular, we do not know of a general way of providing an NP witness for proving the ill-formedness of a public key. We skip this issue and simply assume that it is possible for the time being. We will get back to the issue at the end of the overview. The protocol now proceeds as follows.

1. The prover runs the key generation algorithm of the PKE to obtain a public key pk_P and chooses the first message r'_{zap} of the ZAP system. It then sets the long-term public parameter as $\mathsf{pp} = (\mathsf{pk}_P, r'_{\mathsf{zap}})$.

2. In the first round of the protocol, the verifier chooses the first round message r_{zap} of the ZAP system and a random image $z = f(y)$ of the OWP $f : \{0,1\}^\ell \to \{0,1\}^\ell$. It then runs lossy key generation of the lossy encryption to obtain a public key pk_V. It then proves statement (1) with respect to r'_{zap} by using the randomness for the lossy key generation as a witness to obtain a proof π'_{zap}. Finally, it sends $(r_{\mathsf{zap}}, \mathsf{pk}_V, \pi'_{\mathsf{zap}})$ to the prover.

3. Given the message, the prover verifies π'_{zap} for statement (1) and aborts the protocol if it is not valid. Otherwise, it encrypts the string 0^ℓ under pk_P and pk_V to obtain $\mathsf{ct}_P = \mathsf{PKE.Enc}_{\mathsf{pk}_P}(0^\ell; r_P)$ and $\mathsf{ct}_V = \mathsf{LE.Enc}_{\mathsf{pk}_V}(0^\ell; r_V)$, where LE stands for "lossy encryption". It then proves that there is a witness (w', y', r'_P, r'_V) such that
$$\Big((x, w') \in R\Big) \vee \Big(\mathsf{ct}_P = \mathsf{PKE.Enc}_{\mathsf{pk}_P}(y'; r'_P) \wedge \mathsf{ct}_V = \mathsf{LE.Enc}_{\mathsf{pk}_V}(y'; r'_V) \wedge f(y') = z\Big) \tag{2}$$
with respect to r_{zap} to obtain a proof π_{zap}. Note that in the honest execution, the prover sets $(w', y', r'_P, r'_V) = (w, \bot, \bot, \bot)$. It then sends $\pi = (\mathsf{ct}_P, \mathsf{ct}_V, \pi_{\mathsf{zap}})$ to the verifier.

4. Given the proof from the prover, the verifier parses $\pi \to (\mathsf{ct}_P, \mathsf{ct}_V, \pi_{\mathsf{zap}})$ and verifies the proof π_{zap} with respect to statement (2).

First attempt of the security proof. We now try to prove the security of the scheme. We first prove the zero-knowledge property with a QPT simulator. To do so, we start from the real game where a malicious verifier interacts with an honest prover and gradually change the prover into a zero-knowledge simulator through game hops. In the first step, we change the prover to be a quantum algorithm, which inverts the OWP to recover $y = f^{-1}(z)$ from the first round message by the verifier. We then change the game so that the prover encrypts y instead of 0^ℓ when it generates the ciphertext ct_P. Due to the security of PKE against QPT adversaries, this game is indistinguishable from the real game. In the next step, we replace the ciphertext ct_V with the encryption of y instead of 0^ℓ. We show that this game is indistinguishable from the previous game by combining the soundness of the ZAP system and the lossiness of the lossy encryption. Without loss of generality, we can assume that the prover does not abort the interaction, since otherwise the malicious verifier cannot obtain any information. However, if the prover does not reject the malicious verifier, this means that statement (1) holds by the soundness of the ZAP. Since pk_P is honestly chosen, pk_V should be a lossy key. Then, by the lossiness of the lossy encryption, we conclude that ct_V does not carry any information about the message, and the change does not alter the distribution of ct_V. Finally, we change the game so that the prover uses the latter branch of statement (2) to generate π_{zap}. Due to the witness indistinguishability of the ZAP, this game is indistinguishable from the previous game. Notice that the prover in the final game does not use the witness w for the statement x to generate the proof and thus constitutes a zero-knowledge simulator.

We then proceed to the proof of the soundness. The proof will be by case analysis. In both cases, we construct a non-uniform reduction algorithm that inverts the OWP. First, we consider the case where the malicious prover chooses honestly generated pk_P. In this case, the reduction algorithm receives pk_P from the malicious prover and finds the corresponding secret key sk_P in the pre-computation phase using its unbounded computational power. Then, in the online phase, it receives the problem instance $z = f(y)$ of the OWP and embeds it into the first-round message from the verifier to the prover. If the malicious prover manages to generate an accepting proof for $x \notin \mathcal{L}$, this should satisfy the trapdoor branch of statement (2) by the soundness of the ZAP. In particular, ct_P should be an encryption of $y = f^{-1}(z)$ under the public key pk_P and thus the reduction algorithm can successfully extract y from ct_P by using sk_P.

We next consider the other case where pk_P is ill-formed. In this case, we need a game hop. In the first step, we change the verifier to be a non-uniform algorithm and have it compute the NP witness for the ill-formedness of pk_P.

Then, the verifier generates the proof using the latter branch of statement (1). This game is indistinguishable from the previous game by the witness indistinguishability of the ZAP. In the next step, we change the game so that the verifier generates pk_V by the normal key generation algorithm rather than the lossy key generation algorithm. This game is indistinguishable from the previous game by the security of the lossy encryption. Note that this game hop is possible because the verifier no longer needs the witness that proves pk_V is generated from the lossy key generation due to the change introduced in the previous game. We are now ready to construct the inverter for OWP. Similarly to the case where pk_P is honestly generated, the soundness of the ZAP implies that an accepting proof for $x \notin \mathcal{L}$ satisfies the latter branch of statement (1). This time, the inverter extracts y from ct_V, which is possible because pk_V is now changed to be a normal public key rather than a lossy one.

While the above proof sketch is almost correct, there is still a subtle issue. In particular, the proof of the soundness for the case of ill-formed pk_P is not correct. The problem is that we cannot prove that the winning probability of the malicious prover is changed only negligibly through the game changes because we cannot construct a corresponding reduction algorithm that establishes this. For example, we try to construct a reduction algorithm that breaks the witness indistinguishability of the ZAP by assuming a malicious prover whose success probability in the second game is non-negligibly different from that in the first game. A natural way to do so is to let the reduction algorithm output 1 only when the malicious prover successfully breaks the soundness of our argument system. However, this is not possible since the reduction algorithm cannot efficiently decide whether the output (x^*, π^*) of the malicious prover violates the soundness or not. In particular, even if the malicious prover outputs an accepting pair of a statement x^* and a proof π^*, x^* may be in $\mathcal{L}$ and the reduction algorithm cannot detect it, since $\mathcal{L}$ may be hard to decide language. To address this problem, we further change the protocol.

Making the winning condition efficiently checkable. As we observed above, the only reason why the winning condition is not efficiently checkable is that the language $\mathcal{L}$ is not efficiently decidable in general. To resolve the problem, we change the protocol so that the prover explicitly includes an encrypted version of witness w in the proof. In more details, we change the protocol so that we add a public key $\widehat{\mathsf{pk}}_P$ of another instance of PKE to the public parameter of the prover and change the prover so that it outputs $\widehat{\mathsf{ct}}_P = \mathsf{PKE.Enc}_{\widehat{\mathsf{pk}}_P}(w; \widehat{r}_P)$ along with ct_P and ct_V and proves that there is a witness $(w', \widehat{r}'_{\mathsf{KeyGen}}, \widehat{r}'_P, y', r'_P, r'_V)$ such that

$$\Big((x, w') \in R \wedge \big(\widehat{\mathsf{pk}}_P \text{ is generated by } \mathsf{PKE.KeyGen}(1^\kappa; \widehat{r}'_{\mathsf{KeyGen}})\big) \wedge \widehat{\mathsf{ct}}_P = \mathsf{PKE.Enc}_{\widehat{\mathsf{pk}}_P}(w'; \widehat{r}'_P)\Big)$$
$$\bigvee \Big(\mathsf{ct}_P = \mathsf{PKE.Enc}_{\mathsf{pk}_P}(y'; r'_P) \wedge \mathsf{ct}_V = \mathsf{LE.Enc}_{\mathsf{pk}_V}(y'; r'_V) \wedge f(y') = z\Big), \tag{3}$$

where the former branch is used in the honest execution of the protocol and the latter is for the simulation and is not changed from the previous construction. Note that to prove the former branch, the prover needs randomness $\widehat{r}_{\mathsf{KeyGen}}$ used in the key generation of $\widehat{\mathsf{pk}}_P$ and thus it has to keep the randomness as a secret parameter. This needs to change the syntax of the zero-knowledge argument system again. However, it does not affect the application to blind signatures, since the syntax of the latter allows the prover to have a secret key.

We then explain how the above change helps. In our proof for the soundness, we relax the winning condition so that the adversary is said to semi-win the game if it outputs an accepting proof $\pi^* = (\mathsf{ct}^*_P, \mathsf{ct}^*_V, \widehat{\mathsf{ct}}^*_P, \pi^*_{\mathsf{zap}})$ for x^* and $\widehat{\mathsf{pk}}_P$ is not in the range of the key generation algorithm or $\widehat{\mathsf{ct}}^*_P$ is not an encryption of a witness w^* such that $R(x^*, w^*) = 1$. We observe that to check this modified winning condition, it is unnecessary to perform the membership test of the language $\mathcal{L}$. The modified winning condition is efficiently checkable for the non-uniform reduction algorithm as follows. It first checks whether $\widehat{\mathsf{pk}}_P$ is honestly generated or not in the pre-computation phase and find the corresponding secret key by brute-force search if it is so. Then, in the online phase, it decrypts the ciphertext $\widehat{\mathsf{ct}}^*_P$ using the secret key to see if the decryption result w^* satisfies $R(x^*, w^*) = 1$ or not. We note that since we relaxed the winning condition, the adversary is regarded as (semi-)winning the game even when it outputs an accepting proof for $x^* \in \mathcal{L}$ if it chooses ill-formed $\widehat{\mathsf{pk}}_P$ or $\widehat{\mathsf{ct}}^*_P$ that does not encrypt the witness for x^*. However, these events happen only with negligible probability and thus can be ignored, since these events imply that the soundness of the ZAP is violated.

Certifying invalid public keys. Now, the only remaining problem is how to prove the statement that pk_P is an ill-formed public key. We show that it is possible to provide an NP witness for this statement if we use Regev's PKE

scheme [Reg05, Reg09]. In Regev's PKE scheme, a public key consists of description of a basis of a lattice L and a vector $\mathbf{v}$. The secret key is the vector in L closest to $\mathbf{v}$. For an honestly generated public key, the distance $\mathsf{dist}(L, \mathbf{v})$ between L and $\mathbf{v}$ is close, while for a maliciously generated key, the distance may be far. Therefore, our goal is to provide a proof that $\mathbf{v}$ is far from L. For this purpose, we use the result by Aharonov and Regev [AR04, AR05], who showed that a language consisting of a pair of a lattice and a vector whose distance is far constitutes an NP language. The subtle point is that their proof is for "gap language" in the sense that they cannot give an NP witness for the pair of a lattice and a vector whose distance is neither far enough nor close enough. Translated to our setting, this means that a malicious prover in our zero-knowledge argument system may choose a public key that is not in the support of the honest key generation algorithm without being caught, if the lattice and the vector are not very much far. We show that we can still define a secret key for such a public key that enables the extraction of the message from the ciphertext, which is sufficient for our purpose.

1.4 Related Work

(Im)possibility of round-optimal blind signature. Lindell proved that it is impossible to achieve round-optimal blind signatures in the plain model under the simulation-based security definition [Lin08]. If we do not rely on setup assumptions, the best possible security that round-optimal blind signatures can satisfy is game-based security. Moreover, Fischlin and Schröder [FS10] prove that 3-move (and fewer moves) blind signature schemes cannot be secure under non-interactive assumptions in the plain model via black-box reductions *if they satisfy all the following conditions*: (1) it is 3-move (or fewer moves), (2) it satisfies computational blindness (3) we can efficiently check whether the execution of the scheme yields a valid signature from its transcript, (4) we can efficiently verify whether a verification-key has a corresponding signing key, and (5) its blindness holds relative to a forgery oracle.[9] Pass [Pas11] prove that there is no black-box reduction from round-optimal *unique*[10] blind signatures to non-interactive assumptions. That is, although there are black-box impossibility results on round-optimal blind signatures from standard assumptions in the plain model, there are still possibilities by avoiding the conditions. We circumvent the impossibility result of [FS10] by using quantum reductions whereas their impossibility result is only applicable to schemes with classical reductions. In a little more detail, the work of [FS10] constructs an adversary that breaks the blindness using a reduction algorithm for the unforgeability. In our case, the reduction algorithm for the unforgeability is QPT, so application of their result would yield a QPT adversary that breaks the blindness. This does not contradict blindness against classical PPT adversaries, which we prove in this paper.

Garg, Rao, Sahai, Schröder, and Unruh [GRS+11] presented the first round-optimal blind signature scheme from *standard assumptions in the plain model* by using the complexity leveraging technique. Fuchsbauer, Hanser, and Slamanig [FHS15] presented a practical round-optimal blind signature scheme in the plain model without complexity leveraging by using interactive assumptions.

There are round-optimal blind signature schemes based on non-interactive assumptions in the plain model without complexity leveraging [HK16, DFKS16], but they are secure in the honest-signer model, where a signer is semi-honest and does not deviate from protocols.

Non-round-optimal blind signature. Abe presented a 3-move blind signature scheme from standard assumptions in the RO model [Abe01]. Hazay, Katz, Koo, and Lindell presented a concurrently secure blind signature scheme in the standard model under a game-based security definition, but it is not round-optimal [HKKL07]. Okamoto presented a 4-move blind signature scheme based on q-type assumptions in the plain model [Oka06]. Hauck, Kiltz, and Loss presented a modular framework to construct 3-move blind signature schemes from standard assumptions in the RO model [HKL19].

ZK protocols with less than four moves. Our notion of blind-signature-conforming ZK arguments syntactically can be seen as a delayed-input three-move protocol with a reusable first message.[11] Here, we review known one-/two-/three-move ZK protocols and explain that none of them suffices for our purpose.

[9] If we replace condition (2) with (2') it satisfies statistical blindness, we need neither condition (4) nor (5).

[10] If a message is fixed, a valid signature is uniquely determined.

[11] We say that a ZK argument is delayed-input if a statement and witness are not used until generating the last message.

Bitansky, Khurana, and Paneth [BKP19] constructed a three-move ZK protocol with the weak ZK property based on standard polynomial assumptions.[12] However, their protocol does not satisfy the delayed-input property, and thus that does not suffice for our purpose.

Several works [BP04, JKKR17, BGI+17, BL18] constructed one-/two-move ZK arguments with super-polynomial simulation based on super-polynomial time assumptions.[13] However, these constructions rely on 2-layered complexity leveraging similarly to Pass' protocol [Pas03], and thus they do not suffice for our purpose even if we try to replace super-polynomial security with quantum polynomial-time security using the idea of [KK19]. (See the paragraph "Zero-Knowledge argument system by Pass" in Section 1.3 for details.) Jain et al. [JKKR17] also showed that their protocol can be proven secure only with polynomial-time assumptions if we augment the protocol to a three-move one. This protocol is delayed-input and uses complexity leveraging only once. However, their protocol does not satisfy (super-polynomial simulation) ZK property when the first message is reused, and thus does not suffice for our purpose even with the idea of [KK19].

Bitansky et al. [BBK+16] constructs a three-move ZK protocol with weaker soundness that holds only against uniform adversaries. This does not suffice for our purpose since we require soundness against non-uniform adversaries (even if we only require uniform security for the resulting blind signatures).

Bitansky et al. [BCPR14] constructed a two-move ZK protocol with a weaker zero-knowledge property that holds only against non-uniform adversaries with bounded-size advice. Such a weaker ZK property may suffice if we focus on blind signatures with uniform security.[14] However, their ZK protocol relies on super-polynomial security of a delegation protocol [KRR14]. Since there is no known construction of a delegation protocol with the required property based on a quantum polynomial-time assumption, we cannot apply the idea of [KK19] to reduce the usage of a super-polynomial time assumption.

2 Preliminaries

Notation. For a positive integer n, $[n]$ denotes a set $\{1, ..., n\}$. For a bit string x, $|x|$ denotes its bit-length. For a set S, we write $s \xleftarrow{\$} S$ to denote the operation of sampling a random s from the uniform distribution over S. For a (probabilistic classical or quantum) algorithm $\mathcal{A}$, we write $y \xleftarrow{\$} \mathcal{A}(x)$ to mean that we run $\mathcal{A}$ on input x and the output is y. For a probabilistic classical algorithm $\mathcal{A}$, we write $\mathcal{A}(x; r)$ to mean the output of $\mathcal{A}$ on input x and randomness r. Moreover, by a slight abuse of notation, we write $y \xleftarrow{\$} \mathcal{A}(x; r)$ to mean that we uniformly pick r from the randomness space of $\mathcal{A}$ and then set $y := \mathcal{A}(x; r)$. For a probabilistic classical algorithm $\mathcal{A}$ that takes as input x and randomness r, "$y \in \mathcal{A}(x)$" means $\Pr_r[y' = y : y' \leftarrow \mathcal{A}(x; r)] > 0$. We use PPT and QPT to mean (classical) probabilistic polynomial time and quantum polynomial time.

A Convention on Non-Uniform Adversaries. When we consider the security of cryptographic primitives against non-uniform classical adversaries, we say that an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ is non-uniform PPT if $\mathcal{A}_0$ is a (possibly randomized) unbounded-time algorithm that takes as input the security parameter 1^κ and outputs a string of length $\mathsf{poly}(\kappa)$ and $\mathcal{A}_1$ is PPT. Typically, $\mathcal{A}_0$ and $\mathcal{A}_1$ can be understood respectively as a "pre-computation phase" that outputs a non-uniform advice and an "online phase" that takes as input the advice and a problem instance and outputs a solution. We note that the randomness of $\mathcal{A}$ does not increase the computational power of $\mathcal{A}$ since $\mathcal{A}_0$ can find the best randomness by using its unbounded computational power. We allow $\mathcal{A}$ to be randomized just for convenience for describing the reductions.

2.1 Non-Interactive Commitment

A non-interactive commitment is a PPT algorithm Com that takes as input the security parameter 1^κ and a message $m \in \{0,1\}^*$ and outputs a commitment com. For notational simplicity, we often omit the security parameter from an input of Com when it is clear from the context.

[12] The weak ZK allows a simulator to depend on a distinguisher.

[13] Constructions in [BP04, BL18] also weaken soundness.

[14] We note that our construction satisfies non-uniform security if we assume non-uniform security for all underlying assumptions. See Remark 2.2.

We require the following security requirements for a non-interactive commitment.

Perfect Binding. For any κ, there does not exist $m \neq m'$ and r, r' such that $\mathsf{Com}(1^\kappa, m; r) = \mathsf{Com}(1^\kappa, m'; r')$.

Computational Hiding against QPT Adversary. For any QPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have

$$\left| \Pr\left[\mathsf{coin}' = \mathsf{coin} : \begin{array}{l} (m_0, m_1, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_0(1^\kappa) \\ \mathsf{coin} \xleftarrow{\$} \{0,1\} \\ \mathsf{com} \xleftarrow{\$} \mathsf{Com}(1^\kappa, m_{\mathsf{coin}}) \\ \mathsf{coin}' \xleftarrow{\$} \mathcal{A}_1(\mathsf{st}, \mathsf{com}) \end{array} \right] - 1/2 \right| = \mathsf{negl}(\kappa).$$

Lombardi and Schaeffer [LS19] constructed a perfectly-binding non-interactive commitment based on a public key encryption scheme with perfect correctness, which in turn exists under the LWE assumption [Reg09]. Though they only consider the hiding property against PPT adversaries, the security proof immediately extends to that for the hiding property against QPT adversaries by assuming the LWE assumption against QPT adversaries.

2.2 Public Key Encryption

A public key encryption (PKE) scheme Π_{PKE} for message space $\mathcal{M}$ consists of PPT algorithms $(\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ defined below.

$\mathsf{PKE.KeyGen}(1^\kappa) \to (\mathsf{ek}_{\mathsf{pke}}, \mathsf{dk}_{\mathsf{pke}})$: The key generation algorithm takes as input the security parameter 1^κ and outputs an encryption key $\mathsf{ek}_{\mathsf{pke}}$ and a decryption key $\mathsf{dk}_{\mathsf{pke}}$.

$\mathsf{PKE.Enc}(\mathsf{ek}_{\mathsf{pke}}, m) \to \mathsf{ct}_{\mathsf{pke}}$: The encryption algorithm takes as input an encryption key $\mathsf{ek}_{\mathsf{pke}}$ and a message $m \in \mathcal{M}$ and outputs a ciphertext $\mathsf{ct}_{\mathsf{pke}}$.

$\mathsf{PKE.Dec}(\mathsf{dk}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{pke}}) \to m$ or $\bot$: The decryption algorithm takes as input a decryption key $\mathsf{dk}_{\mathsf{pke}}$ and a ciphertext $\mathsf{ct}_{\mathsf{pke}}$ and outputs a message $m \in \mathcal{M}$ or a special symbol $\bot$ indicating decryption failure.

Perfect Correctness. For any $\kappa \in \mathbb{N}$, $m \in \mathcal{M}$, $(\mathsf{ek}_{\mathsf{pke}}, \mathsf{dk}_{\mathsf{pke}}) \in \mathsf{PKE.KeyGen}(1^\kappa)$, and $\mathsf{ct}_{\mathsf{pke}} \in \mathsf{PKE.Enc}(\mathsf{ek}_{\mathsf{pke}}, m)$, we have $\mathsf{PKE.Dec}(\mathsf{dk}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{pke}}) = m$.

CPA Security against QPT Adversary. For any QPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have

$$\left| \Pr\left[\mathsf{coin}' = \mathsf{coin} : \begin{array}{l} (\mathsf{ek}_{\mathsf{pke}}, \mathsf{dk}_{\mathsf{pke}}) \xleftarrow{\$} \mathsf{PKE.KeyGen}(1^\kappa) \\ (m_0, m_1, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_0(\mathsf{ek}_{\mathsf{pke}}) \\ \mathsf{coin} \xleftarrow{\$} \{0,1\} \\ \mathsf{com} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{ek}_{\mathsf{pke}}, m_{\mathsf{coin}}) \\ \mathsf{coin}' \xleftarrow{\$} \mathcal{A}_1(\mathsf{st}, \mathsf{com}) \end{array} \right] - 1/2 \right| = \mathsf{negl}(\kappa).$$

There exists a PKE scheme that satisfies the above properties assuming the LWE assumption against QPT adversaries [Reg09]. We note that although the PKE of [Reg09] only satisfies correctness with a negligible decryption error, it can be easily modified to satisfy perfect correctness (see Appendix B.2).

2.3 Lossy Encryption

A lossy PKE scheme Π_{LE} for message space $\mathcal{M}$ consists of PPT algorithms $(\mathsf{LE.InjGen}, \mathsf{LE.LossyGen}, \mathsf{LE.Enc}, \mathsf{LE.Dec})$. The difference between a standard PKE is that a lossy PKE comes with two key generation algorithms: $\mathsf{LE.InjGen}$ and $\mathsf{LE.LossyGen}$ defined below.

$\mathsf{LE.InjGen}(1^\kappa) \to (\mathsf{ek}_{\mathsf{le}}, \mathsf{dk}_{\mathsf{le}})$: The injective key generation algorithm takes as input the security parameter 1^κ and outputs an *injective* encryption key $\mathsf{ek}_{\mathsf{le}}$ and a decryption key $\mathsf{dk}_{\mathsf{le}}$.

$\mathsf{LE.LossyGen}(1^\kappa) \to \mathsf{ek}_{\mathsf{le}}$: The key generation algorithm takes as input the security parameter 1^κ and outputs a *lossy* encryption key $\mathsf{ek}_{\mathsf{le}}$.

Perfect Correctness on Injective Keys. For any $\kappa \in \mathbb{N}$, $m \in \mathcal{M}$, $(\mathsf{ek}_{\mathsf{le}}, \mathsf{dk}_{\mathsf{le}}) \in \mathsf{LE.InjGen}(1^\kappa)$, and $\mathsf{ct}_{\mathsf{le}} \in \mathsf{LE.Enc}(\mathsf{ek}_{\mathsf{le}}, m)$, we have $\mathsf{LE.Dec}(\mathsf{dk}_{\mathsf{le}}, \mathsf{ct}_{\mathsf{le}}) = m$.

Indistinguishability of Keys Against Non-Uniform PPT Adversary. For any κ and non-uniform PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have

$$\left| \Pr\left[\mathcal{A}_1(\mathsf{st}, \mathsf{ek}_{\mathsf{le}}) = 1 : \begin{array}{l} \mathsf{st} \xleftarrow{\$} \mathcal{A}_0(1^\kappa) \\ (\mathsf{ek}_{\mathsf{le}}, \mathsf{dk}_{\mathsf{le}}) \xleftarrow{\$} \mathsf{LE.InjGen}(1^\kappa) \end{array} \right] - \Pr\left[\mathcal{A}_1(\mathsf{st}, \mathsf{ek}_{\mathsf{le}}) = 1 : \begin{array}{l} \mathsf{st} \xleftarrow{\$} \mathcal{A}_0(1^\kappa) \\ \mathsf{ek}_{\mathsf{le}} \xleftarrow{\$} \mathsf{LE.LossyGen}(1^\kappa) \end{array} \right] \right| \leq \mathsf{negl}(\kappa).$$

Lossiness of Lossy Keys. For any κ, $\mathsf{ek}_{\mathsf{le}} \in \mathsf{LE.LossyGen}(1^\kappa)$, and $m_0, m_1 \in \mathcal{M}$, the statistical distance between the distributions $\mathsf{LE.Enc}(\mathsf{ek}_{\mathsf{le}}, m_0)$ and $\mathsf{LE.Enc}(\mathsf{ek}_{\mathsf{le}}, m_1)$ is negligible.

There exists a lossy PKE scheme that satisfies the above properties assuming the DDH, DLIN, quadratic residuosity (QR), or decisional composite residuosity (DCR) assumptions against non-uniform PPT adversaries [PVW08, BHY09, HLOV11].

2.4 ZAP

ZAP [DN07] is a public coin 2-move witness indistinguishable non-interactive argument. More precisely, a ZAP system Π_{zap} for an NP language $\mathcal{L}$ corresponding to a relation $\mathcal{R}$ with public coin length ℓ consists of two PPT algorithms $(\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$.

$\mathsf{ZAP.Prove}(r, x, w) \to \pi$**:** The proving algorithm is given the public coin $r \in \{0,1\}^\ell$, a statement x, and a witness w, and outputs a proof π.

$\mathsf{ZAP.Verify}(r, x, \pi) \to \top$ **or** $\bot$**:** The verification algorithm is given the public coin $r \in \{0,1\}^\ell$, a statement x, and a proof π, and outputs $\top$ indicating acceptance or $\bot$ indicating rejection.

Completeness. For any $\kappa \in \mathbb{N}$, $r \in \{0,1\}^\ell$, and $(x, w) \in \mathcal{R}$, we have

$$\Pr[\mathsf{ZAP.Verify}(r, x, \mathsf{ZAP.Prove}(r, x, w)) = \top] = 1.$$

Adaptive Statistical Soundness. For any unbounded-time adversary $\mathcal{A}$, we have

$$\Pr[\mathsf{ZAP.Verify}(r, x^*, \pi) = \top \wedge x^* \notin \mathcal{L} : r \xleftarrow{\$} \{0,1\}^\ell, (x^*, \pi) \xleftarrow{\$} \mathcal{A}(r)] = \mathsf{negl}(\kappa).$$

Adaptive Computational Witness Indistinguishability against Non-Uniform PPT Adversary. For any non-uniform PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have

$$\left| \Pr\left[\begin{array}{l} 1 \xleftarrow{\$} \mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, \pi_0) \\ \wedge\ (x, w_0) \in \mathcal{R}\ \wedge\ (x, w_1) \in \mathcal{R} \end{array} : \begin{array}{l} (r, x, w_0, w_1, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}_0(1^\kappa) \\ \pi_0 \xleftarrow{\$} \mathsf{ZAP.Prove}(r, x, w_0) \end{array} \right] \right.$$

$$-\Pr\left[\begin{array}{ll} 1 \xleftarrow{\$} \mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, \pi_1) & \\ \wedge\ (x, w_0) \in \mathcal{R} \ \wedge\ (x, w_1) \in \mathcal{R} & : \begin{array}{l}(r, x, w_0, w_1, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}_0(1^\kappa) \\ \pi_1 \xleftarrow{\$} \mathsf{ZAP.Prove}(r, x, w_1)\end{array} \end{array}\right] \leq \mathsf{negl}(\kappa).$$

As shown in [DN07], ZAP satisfying the above properties can be constructed based on any NIZK proofs (with zero-knowledge against non-uniform PPT verifiers) in the common random string model, which in turn is known to exist under the (non-uniform PPT) hardness of factoring. [FLS99]. Another instantiation is possible based on SXDH or DLIN assumptions (against non-uniform PPT adversaries) in pairing groups [GOS12]. We note that there is no known construction of ZAP from polynomial-time quantum assumptions.

2.5 Digital Signatures

Here, we give a definition of digital signatures. A digital signature scheme with a message space $\mathcal{M}$ consists of PPT algorithms $(\mathsf{SigGen}, \mathsf{Sign}, \mathsf{SigVerify})$.

$\mathsf{SigGen}(1^\kappa) \to (\mathsf{svk}, \mathsf{ssk})$**:** The key generation algorithm takes as input the security parameter 1^κ and outputs a verification key svk and a signing key ssk.

$\mathsf{Sign}(\mathsf{ssk}, m) \to \sigma$**:** The signing algorithm that takes as input a signing key ssk and a message $m \in \mathcal{M}$ and outputs a signature σ.

$\mathsf{SigVerify}(\mathsf{svk}, m, \sigma) \to \top$ **or** $\bot$**:** The deterministic verification algorithm that takes as input a verification key vk, a message $m \in \mathcal{M}$, and a signature σ, and outputs $\top$ to indicate acceptance or $\bot$ to indicate rejection.

Correctness. For any $\kappa \in \mathbb{N}$, $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{SigVerify}(\mathsf{svk}, m, \sigma) = \bot : \begin{array}{l}(\mathsf{svk}, \mathsf{ssk}) \xleftarrow{\$} \mathsf{SigGen}(1^\kappa) \\ \sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{ssk}, m)\end{array}\right] = \mathsf{negl}(\kappa).$$

Unforgeability against QPT Adversary. For any QPT adversary $\mathcal{A}$ with classical access to a signing oracle $\mathsf{Sign}(\mathsf{ssk}, \cdot)$, we have

$$\Pr\left[\begin{array}{ll} \mathsf{SigVerify}(\mathsf{svk}, m, \sigma) = \top & \\ \wedge\ \mathcal{A} \text{ never queried } m & : \begin{array}{l}(\mathsf{svk}, \mathsf{ssk}) \xleftarrow{\$} \mathsf{SigGen}(1^\kappa) \\ (m, \sigma) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sign}(\mathsf{ssk}, \cdot)}(\mathsf{svk})\end{array} \end{array}\right] = \mathsf{negl}(\kappa)$$

A digital signature scheme that satisfies the above properties can be constructed based on any OWF against quantum adversaries [Rom90]. More direct and efficient constructions exist based on the LWE assumption against QPT adversaries [ABB10, CHKP12].

2.6 Secure Function Evaluation

A secure function evaluation (SFE) is a 2-move protocol between a sender who holds a (classical) circuit C and a receiver who holds x, where the goal is for the receiver to compute $C(x)$ without revealing the inputs to each other. Specifically, SFE consists of PPT algorithms $\Pi_{\mathsf{SFE}} = (\mathsf{Receiver}, \mathsf{Sender}, \mathsf{Derive})$ with the following syntax:

$\mathsf{Receiver}(1^\kappa, x) \to (\mathsf{sfe}_1, \mathsf{sfest})$**:** This is an algorithm supposed to be run by a receiver that takes the security parameter 1^κ and x as input and outputs a first message sfe_1 and a receiver's state sfest.

$\mathsf{Sender}(1^\kappa, \mathsf{sfe}_1, C) \to \mathsf{sfe}_2$**:** This is an algorithm supposed to be run by a sender that takes the security parameter 1^κ, a first message sfe_1 sent from a receiver and a description of a classical circuit C as input and outputs a second message sfe_2.

$\mathsf{Derive}(\mathsf{sfest}, \mathsf{sfe}_2) \to y$**:** This is an algorithm supposed to be run by a receiver that takes a receiver's state sfest and a second message sfe_2 as input and outputs a string y.

Correctness. For any $\kappa \in \mathbb{N}$, C, and x, we have

$$\Pr[\mathsf{Derive}(\mathsf{sfest}, \mathsf{sfe}_2) = C(x) : (\mathsf{sfe}_1, \mathsf{sfest}) \xleftarrow{\$} \mathsf{Receiver}(1^\kappa, x), \mathsf{sfe}_2 \xleftarrow{\$} \mathsf{Sender}(1^\kappa, \mathsf{sfe}_1, C)] = 1.$$

Security requirements are essentially the same as those in [GRS+11] except that we require the extraction algorithm to run in QPT instead of classical super-polynomial time. Specifically, we require the following two security notions.

Receiver's Security against Non-Uniform PPT Adversary. For any pair of inputs (x_0, x_1) and non-uniform PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have

$$\left| \Pr\left[\mathcal{A}_1(\mathsf{st}, \mathsf{sfe}_1) = 1 : \begin{array}{l} \mathsf{st} \xleftarrow{\$} \mathcal{A}_0(1^\kappa) \\ (\mathsf{sfe}_1, \mathsf{sfest}) \xleftarrow{\$} \mathsf{Receiver}(1^\kappa, x_0) \end{array} \right] - \Pr\left[\mathcal{A}_1(\mathsf{st}, \mathsf{sfe}_1) = 1 : \begin{array}{l} \mathsf{st} \xleftarrow{\$} \mathcal{A}_0(1^\kappa) \\ (\mathsf{sfe}_1, \mathsf{sfest}) \xleftarrow{\$} \mathsf{Receiver}(1^\kappa, x_1) \end{array} \right] \right| \leq \mathsf{negl}(\kappa).$$

Quantum-Extraction Sender's Security against QPT Adversary. There exists a QPT algorithm SFEExt and a PPT algorithm SFESim that satisfy the following: For any QPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have

$$\Big|\Pr[\mathcal{A}_1(\mathsf{st}_\mathcal{A}, \mathsf{sfe}_2) = 1 : (\mathsf{sfe}_1, C, \mathsf{st}_\mathcal{A}) \xleftarrow{\$} \mathcal{A}_0(1^\kappa), \mathsf{sfe}_2 \xleftarrow{\$} \mathsf{Sender}(1^\kappa, \mathsf{sfe}_1, C)] - \Pr[\mathcal{A}_1(\mathsf{st}_\mathcal{A}, \mathsf{sfe}_2) = 1 : (\mathsf{sfe}_1, C, \mathsf{st}_\mathcal{A}) \xleftarrow{\$} \mathcal{A}_0(1^\kappa), x \xleftarrow{\$} \mathsf{SFEExt}(\mathsf{sfe}_1), \mathsf{sfe}_2 \xleftarrow{\$} \mathsf{SFESim}(1^\kappa, \mathsf{sfe}_1, C(x))]\Big| \leq \mathsf{negl}(\kappa).$$

An SFE protocol that satisfies these security notions can be constructed based on either of the DDH, QR, or decisional composite residuosity (DCR) assumptions against non-uniform PPT adversaries and LWE assumption against QPT adversaries. Namely, we can construct it based on Yao's 2PC protocol instantiated with secure garbled circuit against quantum adversaries (which can be instantiated based on OWF against quantum adversaries) and non-uniform classical-receiver-secure but quantumly receiver-insecure and statistically sender-private OT (which can be instantiated based on the non-uniform PPT hardness of DDH [NP01], DLIN [LVW20], QR, or DCR [HK12]). See Appendix A for details.

2.7 Blind Signatures

Here, we give a definition of blind signatures. For simplicity, we give a definition focusing on round-optimal blind signatures. A round-optimal blind signature scheme with a message space $\mathcal{M}$ consists of PPT algorithms $(\mathsf{BSGen}, \mathcal{U}_1, \mathcal{S}_2, \mathcal{U}_{\mathsf{der}}, \mathsf{BSVerify})$.

$\mathsf{BSGen}(1^\kappa) \to (\mathsf{pk}, \mathsf{sk})$**:** The key generation algorithm takes as input the security parameter 1^κ and outputs a public key pk and a signing key sk.

$\mathcal{U}_1(\mathsf{pk}, m) \to (\mu, \mathsf{st}_\mathcal{U})$**:** This is the user's first message generation algorithm that takes as input a public key pk and a message $m \in \mathcal{M}$ and outputs a first message μ and a state $\mathsf{st}_\mathcal{U}$.

$\mathcal{S}_2(\mathsf{sk}, \mu) \to \rho$**:** This is the signer's second message generation algorithm that takes as input a signing key sk and a first message μ as input and outputs a second message ρ.

$\mathcal{U}_{\mathsf{der}}(\mathsf{st}_\mathcal{U}, \rho) \to \sigma$**:** This is the user's signature derivation algorithm that takes as input a state $\mathsf{st}_\mathcal{U}$ and a second message ρ as input and outputs a signature σ.

$\mathsf{BSVerify}(\mathsf{pk}, m, \sigma) \to \top$ **or** $\bot$**:** This is a deterministic verification algorithm that takes as input a public key pk, a message $m \in \mathcal{M}$, and a signature σ, and outputs $\top$ to indicate acceptance or $\bot$ to indicate rejection.

Correctness. For any $\kappa \in \mathbb{N}$, $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{BSVerify}(\mathsf{pk}, m, \sigma) = \bot : \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{BSGen}(1^\kappa) \\ (\mu, \mathsf{st}_{\mathcal{U}}) \xleftarrow{\$} \mathcal{U}_1(\mathsf{pk}, m) \\ \rho \xleftarrow{\$} \mathcal{S}_2(\mathsf{sk}, \mu) \\ \sigma \xleftarrow{\$} \mathcal{U}_{\mathsf{der}}(\mathsf{st}_{\mathcal{U}}, \rho)\end{array}\right] = \mathsf{negl}(\kappa).$$

Unforgeability against PPT Adversary. For any $q = \mathsf{poly}(\kappa)$ and PPT adversary $\mathcal{A}$ that makes at most q queries, we have

$$\Pr\left[\begin{array}{l}\mathsf{BSVerify}(\mathsf{pk}, m_i, \sigma_i) = \top \text{ for all } i \in [q+1] \\ \quad \wedge \{m_i\}_{i\in[q+1]} \text{ is pairwise distinct}\end{array} : \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{BSGen}(1^\kappa) \\ \{(m_i, \sigma_i)\}_{i\in[q+1]} \xleftarrow{\$} \mathcal{A}^{\mathcal{S}_2(\mathsf{sk},\cdot)}(\mathsf{pk})\end{array}\right] = \mathsf{negl}(\kappa)$$

where we say that $\{m_i\}_{i\in[q+1]}$ is pairwise distinct if we have $m_i \neq m_j$ for all $i \neq j$.

Blindness against PPT Adversary. For defining blindness, we consider the following game between an adversary $\mathcal{A}$ and a challenger.

Setup. $\mathcal{A}$ is given as input the security parameter 1^κ, and sends a public key pk and a pair of messages (m_0, m_1) to the challenger.

First Message. The challenger generates $(\mu_b, \mathsf{st}_{\mathcal{U},b}) \xleftarrow{\$} \mathcal{U}_1(\mathsf{pk}, m_b)$ for each $b \in \{0,1\}$, picks $\mathsf{coin} \xleftarrow{\$} \{0,1\}$, and gives $(\mu_{\mathsf{coin}}, \mu_{1-\mathsf{coin}})$ to $\mathcal{A}$.

Second Message. The adversary sends $(\rho_{\mathsf{coin}}, \rho_{1-\mathsf{coin}})$ to the challenger.

Signature Derivation. The challenger generates $\sigma_b \xleftarrow{\$} \mathcal{U}_{\mathsf{der}}(\mathsf{st}_{\mathcal{U},b}, \rho_b)$ for each $b \in \{0,1\}$. If $\sigma_0 = \bot$ or $\sigma_1 = \bot$, then the challenger gives $(\bot, \bot)$ to $\mathcal{A}$. Otherwise, it gives (σ_0, σ_1) to $\mathcal{A}$.

Guess. $\mathcal{A}$ outputs its guess coin'

We say that $\mathcal{A}$ wins if $\mathsf{coin} = \mathsf{coin}'$. We say that a blind signature scheme satisfies blindness if for any PPT adversary $\mathcal{A}$, we have

$$\left|\Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}\right| = \mathsf{negl}(\kappa).$$

Remark 2.1. In a definition of blindness for general (not necessarily round-optimal) blind signatures, $\mathcal{A}$ can schedule interactions with two sessions of a user in an arbitrary order. However, as observed in [GRS+11], the order can be fixed as above without loss of generality when we consider round-optimal schemes.

Remark 2.2. The above definition only requires security against uniform PPT adversaries. We can achieve security against non-uniform PPT adversaries if we assume all assumptions used in this paper hold against non-uniform adversaries. We primarily consider security against uniform adversaries to clarify which assumptions should hold against non-uniform adversaries even if our goal is to prove security against uniform PPT adversaries.

3 Preparations

In this section, we introduce two new primitives used in our construction of blind-signature-conforming zero-knowledge argument in Section 4.

3.1 Classical-Hard Quantum-Solvable Hard Problem Generator

A hard problem generator consists of algorithms $\Pi_{\mathsf{HPG}} = (\mathsf{ProbGen}, \mathsf{VerProb}, \mathsf{Solve}, \mathsf{VerSol})$.

$\mathsf{ProbGen}(1^\kappa) \to \mathsf{prob}$: The problem generation algorithm is a PPT algorithm that is given the security parameter 1^κ as input and outputs a problem $\mathsf{prob} \in \{0,1\}^*$.

$\mathsf{VerProb}(1^\kappa, \mathsf{prob}) \to \top$ **or** $\bot$: The problem verification algorithm is a deterministic classical polynomial-time algorithm that is given the security parameter 1^κ and a problem prob and returns $\top$ if it accepts and $\bot$ if it rejects.

$\mathsf{Solve}(\mathsf{prob}) \to \mathsf{sol}$: The solving algorithm is a QPT algorithm that is given a problem prob and returns a solution sol.

$\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol}) \to \top$ **or** $\bot$: The solution verification algorithm is a deterministic classical polynomial-time algorithm that is given an problem prob and a solution sol, and returns $\top$ if it accepts and $\bot$ if it rejects.

We say that Π_{HPG} is *non-uniform-classical-hard quantum-solvable* if it satisfies the following properties.

Quantum Solvability. For any $\mathsf{prob} \in \{0,1\}^*$ such that $\mathsf{VerProb}(1^\kappa, \mathsf{prob}) = \top$, we have

$$\Pr[\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol}) = \bot : \mathsf{sol} \xleftarrow{\$} \mathsf{Solve}(\mathsf{prob})] = \mathsf{negl}(\kappa).$$

Validity of Honestly Generated Problem. For all $\kappa \in \mathbb{N}$, we have

$$\Pr[\mathsf{VerProb}(1^\kappa, \mathsf{prob}) = \top : \mathsf{prob} \xleftarrow{\$} \mathsf{ProbGen}(1^\kappa)] = 1.$$

Non-Uniform Classical Hardness. For any non-uniform PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have

$$\Pr[\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol}) = \top : \mathsf{st} \xleftarrow{\$} \mathcal{A}_0(1^\kappa), \mathsf{prob} \xleftarrow{\$} \mathsf{ProbGen}(1^\kappa), \mathsf{sol} \xleftarrow{\$} \mathcal{A}_1(\mathsf{st}, \mathsf{prob})] = \mathsf{negl}(\kappa).$$

Remark 3.1. HPG can be trivially constructed based on any OWF with an efficiently recognizable range that is uninvertible by non-uniform PPT adversaries and invertible in QPT by considering an image of the function as prob and its preimage as sol. The efficient recognizability of the range is needed since otherwise we cannot implement VerProb that verifies the existence of a solution. Such an OWF with an efficiently recognizable range can be constructed from the RSA assumption or the discrete logarithm assumption over $\mathbb{Z}_p$ for a prime p of a special form as shown by Goldreich, Levin, and Nisan [GLN11]. (Indeed, their construction is length-preserving and injective and thus any bit-string is in the range of the function.) On the other hand, to the best of our knowledge, there is no known construction of such an OWF from the hardness of factoring or DL over more general groups. This is why we introduce the notion of classical-hard quantum-solvable HPG, which can be seen as a relaxed notion of a OWF with an efficiently recognizable range that is secure against non-uniform classical adversaries and invertible in QPT.

Lemma 3.2. *Assuming the non-uniform classical hardness of factoring or discrete logarithm over an efficiently recognizable cyclic group, there exists classical-hard quantum-solvable hard problem generator.*

Proof. This lemma is an easy consequence of Shor's algorithm [Sho94] that solves factoring and discrete logarithm problems in QPT. For completeness, we describe the constructions.

First, we give a construction based on the non-uniform classical hardness of factoring. For clarity, the non-uniform classical hardness of factoring means that for any non-uniform PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have

$$\Pr[\mathcal{A}_1(\mathsf{st}, N) = (P, Q) : \mathsf{st} \xleftarrow{\$} \mathcal{A}_0(1^\kappa), (P, Q) \xleftarrow{\$} \mathcal{P}_\kappa^2, N := PQ]$$

where $\mathcal{P}_\kappa$ the set of all κ-bit primes. Based on this assumption, we construct a classical-hard quantum-solvable hard problem generator as follows:

$\mathsf{ProbGen}(1^\kappa)$: This algorithm generates $(P, Q) \xleftarrow{\$} \mathcal{P}_\kappa^2$ and outputs $\mathsf{prob} := PQ$.

$\mathsf{VerProb}(1^\kappa, \mathsf{prob})$: This algorithm outputs $\top$ if and only if prob is a valid encoding of some positive integer N. Note that this algorithm does not check that N is a product of two primes of equal size since this cannot be done efficiently.

$\mathsf{Solve}(\mathsf{prob})$: Let N be a positive integer encoded by prob. (If prob does not encode such N, this algorithm aborts.) Then it computes the prime factorization of N by using Shor's algorithm, and outputs it as sol.

$\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol})$: This algorithm outputs $\top$ if and only if sol is the prime factorization of prob. We note that this can be done in classical polynomial time by using the AKS primality test [AKS04].

The above construction satisfies quantum solvability since Shor's algorithm succeeds in factoring an integer with overwhelming probability. It satisfies the validity of honestly generated problem since $\mathsf{ProbGen}$ always outputs a positive integer, which passes $\mathsf{VerProb}$. Non-uniform classical hardness directly follows from the assumed hardness of factoring.

Next, we give a discrete-logarithm-based construction. Let $\mathcal{G} = \{\mathbb{G}_\kappa\}_{\kappa\in\mathbb{N}}$ be a family of cyclic groups $\mathbb{G}_\kappa$ of order p_κ with a generator g_κ that is efficiently recognizable, i.e., there exists a PPT algorithm that decides if a given string encodes an element of $\mathbb{G}$. In the following, we omit the subscript κ when that is clear from context for notational simplicity. We assume that discrete logarithm w.r.t. $\mathcal{G}$ is hard against non-uniform classical adversaries. That is, we assume that for any non-uniform PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have

$$\Pr[\mathcal{A}_1(\mathsf{st}, g^x) = x : \mathsf{st} \xleftarrow{\$} \mathcal{A}_0(1^\kappa), x \xleftarrow{\$} \mathbb{Z}_p] = \mathsf{negl}(\kappa).$$

Based on this assumption, we construct a classical-hard quantum-solvable hard problem generator as follows:

$\mathsf{ProbGen}(1^\kappa)$: This algorithm chooses $x \xleftarrow{\$} \mathbb{Z}_p$ and outputs $\mathsf{prob} := g^x$.

$\mathsf{VerProb}(1^\kappa, \mathsf{prob})$: This algorithm outputs $\top$ if and only if prob is a valid encoding of an element of $\mathbb{G}$.

$\mathsf{Solve}(\mathsf{prob})$: Let $h \in \mathbb{G}$ be an element encoded by prob. (If prob does not encode such h, this algorithm aborts.) Then it computes x such that $g^x = y$ by using Shor's algorithm and outputs $\mathsf{sol} := x$.

$\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol})$: This algorithm outputs $\top$ if and only if we have $g^{\mathsf{sol}} = \mathsf{prob}$.

The above construction satisfies quantum solvability since Shor's algorithm succeeds in solving discrete logarithm problem with overwhelming probability. It satisfies the validity of honestly generated problem since $\mathsf{ProbGen}$ always outputs an element of $\mathbb{G}$, which passes $\mathsf{VerProb}$. Non-uniform classical hardness directly follows from the assumed hardness of discrete logarithm problem. □

3.2 Public Key Encryption with Invalid Key Certifiability.

We introduce a new notion for PKE which we call *invalid key certifiability*. Roughly speaking, it requires that for any (malformed) encryption key $\mathsf{ek}_{\mathsf{ikc}}$, there exists a witness for the invalidness of $\mathsf{ek}_{\mathsf{ikc}}$ or otherwise there must exist a corresponding decryption key that can decrypt ciphertexts under $\mathsf{ek}_{\mathsf{ikc}}$.

More precisely, a PKE scheme $\Pi_{\mathsf{IKC}} = (\mathsf{IKC.KeyGen}, \mathsf{IKC.Enc}, \mathsf{IKC.Dec})$ has *invalid key certifiability* if it additionally has a deterministic classical polynomial-time algorithm $\mathsf{IKC.InvalidVerf}$ with the following syntax and properties:

$\mathsf{IKC.InvalidVerf}(1^\kappa, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{wit}_{\mathsf{invalid}}) \to \top$ **or** $\bot$**:** This algorithm takes the security parameter 1^κ, an encryption key $\mathsf{ek}_{\mathsf{ikc}}$ and a witness $\mathsf{wit}_{\mathsf{invalid}} \in \{0,1\}^\ell$ as input where $\ell(\kappa) = \mathsf{poly}(\kappa)$ is a parameter fixed by the scheme, and outputs $\top$ or $\bot$.

We require the following two properties:

1. For any $\kappa \in \mathbb{N}$ and $(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{dk}_{\mathsf{ikc}}) \xleftarrow{\$} \mathsf{IKC.KeyGen}(1^\kappa)$, there does not exist $\mathsf{wit}_{\mathsf{invalid}} \in \{0,1\}^\ell$ such that $\mathsf{IKC.InvalidVerf}(1^\kappa, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{wit}_{\mathsf{invalid}}) = \top$.
2. For any $\kappa \in \mathbb{N}$ and (possibly malformed) $\mathsf{ek}_{\mathsf{ikc}}$, if there does not exist $\mathsf{wit}_{\mathsf{invalid}} \in \{0,1\}^\ell$ such that $\mathsf{IKC.InvalidVerf}(1^\kappa, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{wit}_{\mathsf{invalid}}) = \top$, then there exists $\mathsf{dk}_{\mathsf{ikc}}$ such that for any m, we have
$$\Pr[\mathsf{IKC.Dec}(\mathsf{dk}_{\mathsf{ikc}}, \mathsf{IKC.Enc}(\mathsf{ek}_{\mathsf{ikc}}, m)) = m] = 1.$$
We call such $\mathsf{dk}_{\mathsf{ikc}}$ a corresponding decryption key to $\mathsf{ek}_{\mathsf{ikc}}$. We say that $\mathsf{ek}_{\mathsf{ikc}}$ is undecryptable if there does not exist a corresponding decryption key to $\mathsf{ek}_{\mathsf{ikc}}$.

Remark 3.3. Remark that we do not require the converse of Item 2, i.e., we do not require that "if there exists $\mathsf{wit}_{\mathsf{invalid}} \in \{0,1\}^{\ell}$ such that $\mathsf{IKC.InvalidVerf}(1^{\kappa}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{wit}_{\mathsf{invalid}}) = \top$, then $\mathsf{ek}_{\mathsf{ikc}}$ is undecryptable". That is, even if $\mathsf{ek}_{\mathsf{ikc}}$ has a corresponding decryption key, it may also have a witness for the invalidness.

Remark 3.4. All dense PKE schemes, in which any string can be a valid encryption key, satisfy invalid key certifiability since all bit strings can be a valid encryption key that has a corresponding decryption key. However, a PKE scheme with invalid key certifiability may not be dense. We note that there is no known candidate of a dense PKE scheme against quantum adversaries.

Lemma 3.5. *There exists a PKE scheme that satisfies the CPA security against QPT adversaries and invalid key certifiability under the quantum hardness of LWE problem.*

The construction is almost identical to the Regev's PKE scheme [Reg09] (modulo some tweak in the parameter). To show the invalid key certifiability property, we rely on the result that the (approximated) gap closest vector (GapCVP) problem lies in $\mathbf{NP} \cap \mathbf{CoNP}$ [AR05]. In particular, $\mathsf{wit}_{\mathsf{invalid}}$ will be a witness to a NO instance of the GapCVP problem. Then, Item 1 follows since a valid public key of Regev's PKE scheme can be seen as an YES instance to the GapCVP problem and there will exist no witness to prove otherwise (i.e., $\mathsf{wit}_{\mathsf{invalid}}$ does not exist). On the other hand, to show Item 2, we rely on the fact that if the public key is *not* a NO instance to the GapCVP problem, then it is still a public key that admits a "good enough" decryption key (i.e., a short vector slightly larger than an honestly generated one). We refer the full details to Appendix B.

4 Blind-Signature-Conforming Zero-Knowledge Argument

In this section, we define blind-signature-conforming zero-knowledge arguments that are sufficient to construct round-optimal blind signatures and construct it based on standard assumptions. Roughly speaking, a blind-signature-conforming zero-knowledge argument is an interactive argument protocol that satisfies the following properties:

1. publicly verifiable[15] and 2-move with reusable setup by the prover,[16]
2. adaptive soundness with untrusted setup against classical prover, and
3. reusable quantum-simulation zero-knowledge against classical verifier.

4.1 Definition

Let $\mathcal{L}$ be an NP language and $\mathcal{R}$ be the corresponding relation. A blind-signature-conforming zero-knowledge argument for $\mathcal{L}$ has the following syntax:

$\mathsf{Setup}(1^{\kappa}) \to (\mathsf{pp}, \mathsf{sp})$**:** This is a setup algorithm (supposed to be run by a prover) that takes as input the security parameter 1^{κ} and outputs a public parameter pp and a secret parameter sp.

$\mathcal{V}_1(\mathsf{pp}) \to \mathsf{ch}$**:** This is the verifier's first message generation algorithm that takes as input a public parameter pp and outputs a first message ch referred to as a *challenge*.

$\mathcal{P}_2(\mathsf{sp}, \mathsf{ch}, x, w) \to \mathsf{resp}$**:** This is the prover's second message generation algorithm that takes as input a secret parameter sp, a challenge ch, a statement x, and a witness w, and outputs a second message resp referred to as a *response*.

$\mathcal{V}_{\mathsf{out}}(\mathsf{pp}, \mathsf{ch}, x, \mathsf{resp}) \to \top$ **or** $\bot$**:** This is the verification algorithm that takes a public parameter pp, a challenge ch, a statement x, and a response resp, and outputs $\top$ to indicate acceptance or $\bot$ to indicate rejection.

It should satisfy the following properties:

[15] Actually, the public verifiability is not needed in the construction of our blind signatures. We only require this because our construction satisfies this.

[16] We can also view it as a three-move protocol by considering the setup as the prover's first message. However, since the first message is reusable, we view the protocol as a two-move protocol with reusable setup.

Completeness. For any $(x, w) \in \mathcal{R}$, we have

$$\Pr[\mathcal{V}_{\mathsf{out}}(\mathsf{pp}, \mathsf{ch}, x, \mathsf{resp}) = \top : (\mathsf{pp}, \mathsf{sp}) \xleftarrow{\$} \mathsf{Setup}(1^\kappa), \mathsf{ch} \xleftarrow{\$} \mathcal{V}_1(\mathsf{pp}), \mathsf{resp} \xleftarrow{\$} \mathcal{P}_2(\mathsf{sp}, \mathsf{ch}, x, w)] = 1.$$

Adaptive Soundness with Untrusted Setup against Non-Uniform PPT Adversary. For any non-uniform PPT cheating prover $\mathcal{P}^* = (\mathcal{P}^*_{\mathsf{Setup}}, \mathcal{P}^*_2)$, we have

$$\Pr\left[\begin{array}{l} \mathcal{V}_{\mathsf{out}}(\mathsf{pp}, \mathsf{ch}, x^*, \mathsf{resp}) = \top \\ \quad \wedge\ x^* \notin \mathcal{L} \end{array} : \begin{array}{l} (\mathsf{pp}, \mathsf{st}_{\mathcal{P}^*}) \xleftarrow{\$} \mathcal{P}^*_{\mathsf{Setup}}(1^\kappa), \\ \mathsf{ch} \xleftarrow{\$} \mathcal{V}_1(\mathsf{pp}), \\ (x^*, \mathsf{resp}) \xleftarrow{\$} \mathcal{P}^*_2(\mathsf{st}_{\mathcal{P}^*}, \mathsf{ch}) \end{array}\right] \leq \mathsf{negl}(\kappa).$$

Reusable Quantum-Simulation Zero-Knowledge against PPT Adversary. Roughly speaking, we require that there exists a QPT simulator that simulates a view of a PPT cheating verifier that interacts with an honest prover even if the setup is reused many times.

More precisely, there exists a QPT simulator $\mathcal{S}$ such that for any PPT adversary $\mathcal{A}$, we have

$$\left|\Pr\left[\mathcal{A}^{\mathcal{O}_{\mathsf{real}}}(\mathsf{pp}) = 1 : (\mathsf{pp}, \mathsf{sp}) \xleftarrow{\$} \mathsf{Setup}(1^\kappa)\right] - \Pr\left[\mathcal{A}^{\mathcal{O}_{\mathsf{sim}}}(\mathsf{pp}) = 1 : (\mathsf{pp}, \mathsf{sp}) \xleftarrow{\$} \mathsf{Setup}(1^\kappa)\right]\right| \leq \mathsf{negl}(\kappa)$$

where oracles $\mathcal{O}_{\mathsf{real}}$ and $\mathcal{O}_{\mathsf{sim}}$ are defined as follows:

$\underline{\mathcal{O}_{\mathsf{real}}(\mathsf{ch}, x, w)}$
If $(x, w) \in \mathcal{R}$
  Return $\mathsf{resp} \xleftarrow{\$} \mathcal{P}_2(\mathsf{sp}, \mathsf{ch}, x, w)$
Else
  Return $\bot$

$\underline{\mathcal{O}_{\mathsf{sim}}(\mathsf{ch}, x, w)}$
If $(x, w) \in \mathcal{R}$
  Return $\mathsf{resp} \xleftarrow{\$} \mathcal{S}(\mathsf{pp}, \mathsf{ch}, x, 1^{|w|})$
Else
  Return $\bot$

4.2 Construction

Let $\mathcal{L}$ be an NP language and $\mathcal{R}$ be its corresponding relation (i.e., $x \in \mathcal{L}$ if and only if there exists w such that $(x, w) \in \mathcal{R}$). We construct a blind-signature-conforming zero-knowledge argument for $\mathcal{L}$ based on the following building blocks.

- A PKE scheme $\Pi_{\mathsf{PKE}} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ that is CPA secure against QPT adversaries.
- A PKE scheme with invalid key certifiability $\Pi_{\mathsf{IKC}} = (\mathsf{IKC.KeyGen}, \mathsf{IKC.Enc}, \mathsf{IKC.Dec}, \mathsf{IKC.InvalidVerf})$ that is CPA secure against QPT adversaries.
- A lossy PKE scheme $\Pi_{\mathsf{LE}} = (\mathsf{LE.InjGen}, \mathsf{LE.LossyGen}, \mathsf{LE.Enc}, \mathsf{LE.Dec})$ that satisfies key indistinguishability against non-uniform PPT adversaries.
- A classical-hard quantum-solvable hard problem generator $\Pi_{\mathsf{HPG}} = (\mathsf{ProbGen}, \mathsf{VerProb}, \mathsf{Solve}, \mathsf{VerSol})$.
- A ZAP system $\Pi_{\mathsf{zap}} = (\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$ for the NP language $\widetilde{\mathcal{L}} = \widetilde{\mathcal{L}}_1 \cup \widetilde{\mathcal{L}}_2$ that satisfies completeness, adaptive statistical soundness, and adaptive computational witness indistinguishability against non-uniform PPT adversaries where languages $\widetilde{\mathcal{L}}_1$ and $\widetilde{\mathcal{L}}_2$ are defined as follows.

  1. $(x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}) \in \widetilde{\mathcal{L}}_1$ if there exists $(w, \mathsf{dk}_{\mathsf{pke}}, r_{\mathsf{pke\text{-}gen}}, r_{\mathsf{pke\text{-}enc}})$ such that

$$(x, w) \in \mathcal{L},$$

$$(\mathsf{ek}_{\mathsf{pke}}, \mathsf{dk}_{\mathsf{pke}}) = \mathsf{PKE.KeyGen}(1^\kappa; r_{\mathsf{pke\text{-}gen}}),$$

$$\mathsf{ct}_{\mathsf{pke}} = \mathsf{PKE.Enc}(\mathsf{ek}_{\mathsf{pke}}, w; r_{\mathsf{pke\text{-}enc}}).$$

2. $(x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}) \in \widetilde{\mathcal{L}}_2$ if there exists $(\mathsf{sol}, r_{\mathsf{ikc\text{-}enc}}, r_{\mathsf{le\text{-}enc}})$ such that

$$\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol}) = \top,$$

$$\mathsf{ct}_{\mathsf{ikc}} = \mathsf{IKC.Enc}(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{sol}; r_{\mathsf{ikc\text{-}enc}}),$$

$$\mathsf{ct}_{\mathsf{le}} = \mathsf{LE.Enc}(\mathsf{ek}_{\mathsf{le}}, \mathsf{sol}; r_{\mathsf{le\text{-}enc}}).$$

- A ZAP system $\Pi'_{\mathsf{zap}} = (\mathsf{ZAP.Prove}', \mathsf{ZAP.Verify}')$ for the NP language $\widetilde{\mathcal{L}}' = \widetilde{\mathcal{L}}'_1 \cup \widetilde{\mathcal{L}}'_2$ that satisfies completeness, adaptive statistical soundness, and adaptive computational witness indistinguishability against non-uniform PPT adversaries where languages $\widetilde{\mathcal{L}}'_1$ and $\widetilde{\mathcal{L}}'_2$ are defined as follows.
  1. $(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}) \in \widetilde{\mathcal{L}}'_1$ if there exists $r_{\mathsf{le\text{-}gen}}$ such that $\mathsf{ek}_{\mathsf{le}} = \mathsf{LE.LossyGen}(1^\kappa; r_{\mathsf{le\text{-}gen}})$.
  2. $(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}) \in \widetilde{\mathcal{L}}'_2$ if there exists $\mathsf{wit}_{\mathsf{invalid}}$ such that $\mathsf{IKC.InvalidVerf}(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{wit}_{\mathsf{invalid}}) = \top$.

We assume that the first message spaces of Π_{zap} and Π'_{zap} are $\{0,1\}^\ell$, which can be assumed without loss of generality by taking ℓ as an arbitrarily large polynomial in κ. Then our blind-signature-conforming zero-knowledge argument $(\mathsf{Setup}, \mathcal{V}_1, \mathcal{P}_2, \mathcal{V}_{\mathsf{out}})$ is described as follows:

$\mathsf{Setup}(1^\kappa)$**:** The setup algorithm is given the security parameter 1^κ, and works as follows.

1. Generate $(\mathsf{ek}_{\mathsf{pke}}, \mathsf{dk}_{\mathsf{pke}}) := \mathsf{PKE.KeyGen}(1^\kappa; r_{\mathsf{pke\text{-}gen}})$.
2. Generate $(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{dk}_{\mathsf{ikc}}) \xleftarrow{\$} \mathsf{IKC.KeyGen}(1^\kappa)$.
3. Generate $r'_{\mathsf{zap}} \xleftarrow{\$} \{0,1\}^\ell$
4. Output $\mathsf{pp} := (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}})$ and $\mathsf{sp} := (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}}, \mathsf{dk}_{\mathsf{pke}}, r_{\mathsf{pke\text{-}gen}})$.

$\mathcal{V}_1(\mathsf{pp})$**:** The verifier is given a public parameter $\mathsf{pp} = (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}})$, and works as follows.

1. Generate $r_{\mathsf{zap}} \xleftarrow{\$} \{0,1\}^\ell$.
2. Generate $\mathsf{prob} \xleftarrow{\$} \mathsf{ProbGen}(1^\kappa)$.
3. Generate $\mathsf{ek}_{\mathsf{le}} \xleftarrow{\$} \mathsf{LE.LossyGen}(1^\kappa; r_{\mathsf{le\text{-}gen}})$.
4. Generate $\pi'_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), r_{\mathsf{le\text{-}gen}})$.
5. Output $\mathsf{ch} := (r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}})$.

$\mathcal{P}_2(\mathsf{sp}, \mathsf{ch}, x, w)$**:** The prover is given a secret parameter $\mathsf{sp} := (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}}, \mathsf{dk}_{\mathsf{pke}}, r_{\mathsf{pke\text{-}gen}})$, a challenge $\mathsf{ch} = (r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}})$, a statement x, and a witness w, and works as follows.

1. Immediately abort and output $\bot$ if $\mathsf{VerProb}(1^\kappa, \mathsf{prob}) = \bot$ or $\mathsf{ZAP.Verify}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \pi'_{\mathsf{zap}}) = \bot$.
2. Generate $\mathsf{ct}_{\mathsf{ikc}} \xleftarrow{\$} \mathsf{IKC.Enc}(\mathsf{ek}_{\mathsf{ikc}}, 0^{|\mathsf{sol}|})$ and $\mathsf{ct}_{\mathsf{le}} \xleftarrow{\$} \mathsf{LE.Enc}(\mathsf{ek}_{\mathsf{le}}, 0^{|\mathsf{sol}|})$.
3. Generate $\mathsf{ct}_{\mathsf{pke}} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{ek}_{\mathsf{pke}}, w; r_{\mathsf{pke\text{-}enc}})$.
4. Generate $\pi_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}(r_{\mathsf{zap}}, (x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}), (w, \mathsf{dk}_{\mathsf{pke}}, r_{\mathsf{pke\text{-}gen}}, r_{\mathsf{pke\text{-}enc}}))$.
5. Output $\mathsf{resp} := (\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}})$.

$\mathcal{V}_{\mathsf{out}}(\mathsf{pp}, \mathsf{ch}, x, \mathsf{resp})$**:** The verifier is given a public parameter $\mathsf{pp} = (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}})$, a challenge $\mathsf{ch} = (r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}})$, a statement x, and a response $\mathsf{resp} = (\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}})$, and works as follows.

1. Output $\mathsf{ZAP.Verify}(r_{\mathsf{zap}}, (x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}), \pi_{\mathsf{zap}})$.

The correctness of the scheme immediately follows from the correctness of Π_{zap} and Π'_{zap} and the validity of an honestly generated instance of Π_{HPG}.

4.3 Security

In this section, we prove that the construction of zero-knowledge arguments given in Section 4.2 satisfies security defined in Section 4.1.

Soundness.

Theorem 4.1. *If* Π_{IKC} *satisfies invalid key certifiability,* Π_{LE} *satisfies key indistinguishability against non-uniform PPT adversaries,* Π_{HPG} *is hard against non-uniform PPT adversaries,* Π_{ZAP} *satisfies adaptive statistical soundness, and* Π'_{ZAP} *satisfies adaptive computational witness indistinguishability against non-uniform PPT adversaries, then* Π_{ZK} *satisfies the adaptive soundness with untrusted setup against non-uniform PPT adversaries.*

Proof. Let $\mathcal{P}^* = (\mathcal{P}^*_{\mathsf{Setup}}, \mathcal{P}^*_2)$ be a non-uniform PPT cheating prover against adaptive soundness of Π_{ZK}. We consider the following sequence of games between $\mathcal{P}^*$ and a challenger.

Game 1**:** In this game, the challenger simulates the real verifier for $\mathcal{P}^*$. That is, this game proceeds as follows:

1. $\mathcal{P}^*$ generates $(\mathsf{pp} = (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}}), \mathsf{st}_{\mathcal{P}^*}) \xleftarrow{\$} \mathcal{P}^*_{\mathsf{Setup}}(1^\kappa)$ and sends pp to the challenger.
2. The challenger generates $r_{\mathsf{zap}} \xleftarrow{\$} \{0,1\}^\ell$, $\mathsf{prob} \xleftarrow{\$} \mathsf{ProbGen}(1^\kappa)$, $\mathsf{ek}_{\mathsf{le}} \xleftarrow{\$} \mathsf{LE.LossyGen}(1^\kappa; r_{\mathsf{le\text{-}gen}})$, $\pi'_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), r_{\mathsf{le\text{-}gen}})$, and sends $\mathsf{ch} := (r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}})$ to $\mathcal{P}^*$.
3. $\mathcal{P}^*$ generates $(x^*, \mathsf{resp} = (\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}})) \xleftarrow{\$} \mathcal{P}^*_2(\mathsf{st}_{\mathcal{P}^*}, \mathsf{ch})$ and outputs it.

We say that $\mathcal{P}^*$ wins if the verifier returns $\top$ on a statement $x^* \notin \mathcal{L}$ and denote the event that $\mathcal{P}^*$ wins by Win_1. Our goal is to prove $\Pr[\mathsf{Win}_1] = \mathsf{negl}(\kappa)$.

For proving this, we define a relaxed winning condition as follows: We say that $\mathcal{P}^*$ semi-wins if the verifier returns $\top$ and either of the following is satisfied:

1. $\mathsf{ek}_{\mathsf{pke}}$ is not well-formed, i.e., there do not exist $\mathsf{dk}_{\mathsf{pke}}$ and $r_{\mathsf{pke\text{-}gen}}$ such that $(\mathsf{ek}_{\mathsf{pke}}, \mathsf{dk}_{\mathsf{pke}}) = \mathsf{PKE.KeyGen}(1^\kappa; r_{\mathsf{pke\text{-}gen}})$, or
2. $\mathsf{ek}_{\mathsf{pke}}$ is well-formed and $\mathsf{ct}_{\mathsf{pke}}$ does not encrypt a valid witness, i.e., there exist $\mathsf{dk}_{\mathsf{pke}}$ and $r_{\mathsf{pke\text{-}gen}}$ such that $(\mathsf{ek}_{\mathsf{pke}}, \mathsf{dk}_{\mathsf{pke}}) = \mathsf{PKE.KeyGen}(1^\kappa; r_{\mathsf{pke\text{-}gen}})$ and $(x, \mathsf{PKE.Dec}(\mathsf{dk}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{pke}})) \notin \mathcal{R}$.

Let Win'_1 be the event that $\mathcal{P}^*$ semi-wins. When Win_1 occurs, Win'_1 also occurs since if $x^* \notin \mathcal{L}$ and condition 1 is not satisfied, then condition 2 must be satisfied since there does not exist a valid witness for x^*. Therefore, it suffices to prove $\Pr[\mathsf{Win}'_1] = \mathsf{negl}(\kappa)$.[17]

Game 2**:** This game is identical to the previous game except that π'_{zap} is sometimes generated differently. Specifically, the challenger checks immediately after receiving pp from $\mathcal{P}^*$ if there exists a witness $\mathsf{wit}_{\mathsf{invalid}}$ that $\mathsf{ek}_{\mathsf{ikc}}$ is invalid (i.e., $\mathsf{IKC.InvalidVerf}(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{wit}_{\mathsf{invalid}}) = \top$) by a brute-force search. We denote by $\mathsf{Invalid}$ the event that such $\mathsf{wit}_{\mathsf{invalid}}$ exists. If $\mathsf{Invalid}$ occurs, then the challenger generates π'_{zap} as $\pi'_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \mathsf{wit}_{\mathsf{invalid}})$. Otherwise, the challenger works in exactly the same way as in the previous game. We denote by Win'_2 the event that $\mathcal{P}^*$ semi-wins in this game.

Game 3**:** This game is identical to the previous game except that when $\mathsf{Invalid}$ occurs, $\mathsf{ek}_{\mathsf{le}}$ is generated as $(\mathsf{ek}_{\mathsf{le}}, \mathsf{dk}_{\mathsf{le}}) \xleftarrow{\$} \mathsf{LE.InjGen}(1^\kappa)$. Note that nothing is modified if $\mathsf{Invalid}$ does not occur. We denote by Win'_3 the event that $\mathcal{P}^*$ semi-wins in this game.

Lemma 4.2. *If* Π'_{zap} *satisfies adaptive computational witness indistinguishability against non-uniform PPT adversaries, then we have* $\left|\Pr[\mathsf{Win}'_2] - \Pr[\mathsf{Win}'_1]\right| = \mathsf{negl}(\kappa)$.

[17] The motivation of introducing the event Win'_1 is to make the winning condition falsifiable (with unbounded-time pre-computation after seeing $\mathsf{ek}_{\mathsf{pke}}$) so that we can argue that the winning probability changes negligibly through the game hops.

Proof. Since Game 1 and Game 2 are identical when Invalid does not occur, we have $\left|\Pr[\mathsf{Win}'_2] - \Pr[\mathsf{Win}'_1]\right| = \left|\Pr[\mathsf{Win}'_2 \wedge \mathsf{Invalid}] - \Pr[\mathsf{Win}'_1 \wedge \mathsf{Invalid}]\right|$. We prove that this is negligible by considering a non-uniform PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ against the adaptive computational witness indistinguishability of Π'_{zap} as follows:

$\mathcal{A}_0(1^\kappa)$**:** This is an unbounded-time pre-computation phase of $\mathcal{A}$ that works as follows: It runs $(\mathsf{pp} = (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}}), \mathsf{st}_{\mathcal{P}^*}) \xleftarrow{\$} \mathcal{P}^*_{\mathsf{Setup}}(1^\kappa)$, checks if Invalid occurs by a brute-force search, and immediately aborts if Invalid does not occur. Otherwise, let $\mathsf{wit}_{\mathsf{invalid}}$ be a string that satisfies $\mathsf{IKC.InvalidVerf}(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{wit}_{\mathsf{invalid}}) = \top$ (which must exist if Invalid occurs), and searches $\mathsf{dk}_{\mathsf{pke}}$ and $r_{\mathsf{pke\text{-}gen}}$ such that $(\mathsf{ek}_{\mathsf{pke}}, \mathsf{dk}_{\mathsf{pke}}) = \mathsf{PKE.KeyGen}(1^\kappa; r_{\mathsf{pke\text{-}gen}})$ by a brute-force search. If such $\mathsf{dk}_{\mathsf{pke}}$ and $r_{\mathsf{pke\text{-}gen}}$ do not exist, it defines them to be $\bot$. Then it generates $r_{\mathsf{zap}} \xleftarrow{\$} \{0,1\}^\ell$, $\mathsf{prob} \xleftarrow{\$} \mathsf{ProbGen}(1^\kappa)$, and $\mathsf{ek}_{\mathsf{le}} \xleftarrow{\$} \mathsf{LE.LossyGen}(1^\kappa; r_{\mathsf{le\text{-}gen}})$, and outputs r'_{zap}, a statement $x' := (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}})$, witnesses $w'_0 := r_{\mathsf{le\text{-}gen}}$ and $w'_1 := \mathsf{wit}_{\mathsf{invalid}}$, and a state $\mathsf{st}_{\mathcal{A}} := (\mathsf{st}_{\mathcal{P}^*}, r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \mathsf{dk}_{\mathsf{pke}})$.

$\mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, \pi'_{\mathsf{zap}})$**:** This is a PPT online phase of $\mathcal{A}$ that works as follows: It parses $(\mathsf{st}_{\mathcal{P}^*}, r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \mathsf{dk}_{\mathsf{pke}}) \leftarrow \mathsf{st}_{\mathcal{A}}$, sets $\mathsf{ch} := (r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}})$, runs $(x^*, \mathsf{resp}) \xleftarrow{\$} \mathcal{P}^*_2(\mathsf{st}_{\mathcal{P}^*}, \mathsf{ch})$, and parses $(\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}}) \leftarrow \mathsf{resp}$. It outputs 1 if

$$\mathsf{ZAP.Verify}(r_{\mathsf{zap}}, (x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}), \pi_{\mathsf{zap}}) = \top$$

and either of the following is satisfied:

1. $\mathsf{dk}_{\mathsf{pke}} = \bot$, or
2. $\mathsf{dk}_{\mathsf{pke}} \neq \bot$ and $(x, \mathsf{PKE.Dec}(\mathsf{dk}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{pke}})) \notin \mathcal{R}$.

We can see that $\mathcal{A}$ perfectly simulates Game 1 (resp. Game 2) for $\mathcal{P}^*$ if π'_{zap} is generated by using the witness w'_0 (resp. w'_1) conditioned on that Invalid occurs. Moreover $\mathcal{A}$ returns 1 if and only if Invalid occurs and $\mathcal{P}^*$ semi-wins in the corresponding game. Therefore, $\mathcal{A}$'s advantage to break the witness indistinguishability of Π'_{zap} is equal to $\left|\Pr[\mathsf{Win}'_2 \wedge \mathsf{Invalid}] - \Pr[\mathsf{Win}'_1 \wedge \mathsf{Invalid}]\right|$. By the witness indistinguishability of Π'_{zap}, we conclude $\left|\Pr[\mathsf{Win}'_2 \wedge \mathsf{Invalid}] - \Pr[\mathsf{Win}'_1 \wedge \mathsf{Invalid}]\right| = \mathsf{negl}(\kappa)$. □

Lemma 4.3. *If Π_{LE} satisfies key indistinguishability against non-uniform PPT adversaries, then we have $|\Pr[\mathsf{Win}'_3] - \Pr[\mathsf{Win}'_2]| = \mathsf{negl}(\kappa)$.*

Proof. Since Game 2 and Game 3 are identical when Invalid does not occur, we have $\left|\Pr[\mathsf{Win}'_3] - \Pr[\mathsf{Win}'_2]\right| = \left|\Pr[\mathsf{Win}'_3 \wedge \mathsf{Invalid}] - \Pr[\mathsf{Win}'_2 \wedge \mathsf{Invalid}]\right|$. We prove that this is negligible by considering a non-uniform PPT adversary $\mathcal{A}$ against the indistinguishability of keys of Π_{LE} as follows:

$\mathcal{A}_0(1^\kappa)$**:** This is an unbounded-time pre-computation phase of $\mathcal{A}$ that works as follows: It runs $(\mathsf{pp} = (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}}), \mathsf{st}_{\mathcal{P}^*}) \xleftarrow{\$} \mathcal{P}^*_{\mathsf{Setup}}(1^\kappa)$, checks if Invalid occurs by a brute-force search, and immediately aborts if Invalid does not occur. Otherwise, let $\mathsf{wit}_{\mathsf{invalid}}$ be a string that satisfies $\mathsf{IKC.InvalidVerf}(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{wit}_{\mathsf{invalid}}) = \top$ (which must exist if Invalid occurs), and searches $\mathsf{dk}_{\mathsf{pke}}$ and $r_{\mathsf{pke\text{-}gen}}$ such that $(\mathsf{ek}_{\mathsf{pke}}, \mathsf{dk}_{\mathsf{pke}}) = \mathsf{PKE.KeyGen}(1^\kappa; r_{\mathsf{pke\text{-}gen}})$ by a brute-force search. If such $\mathsf{dk}_{\mathsf{pke}}$ and $r_{\mathsf{pke\text{-}gen}}$ do not exist, it defines them to be $\bot$. Then it generates $r_{\mathsf{zap}} \xleftarrow{\$} \{0,1\}^\ell$, and $\mathsf{prob} \xleftarrow{\$} \mathsf{ProbGen}(1^\kappa)$, and outputs $\mathsf{st}_{\mathcal{A}} := (\mathsf{st}_{\mathcal{P}^*}, r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{wit}_{\mathsf{invalid}}, \mathsf{dk}_{\mathsf{pke}})$.

$\mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, \mathsf{ek}_{\mathsf{le}})$**:** This is a PPT online phase of $\mathcal{A}$ that works as follows: It parses $(\mathsf{st}_{\mathcal{P}^*}, r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{wit}_{\mathsf{invalid}}, \mathsf{dk}_{\mathsf{pke}}) \leftarrow \mathsf{st}_{\mathcal{A}}$, generates $\pi'_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \mathsf{wit}_{\mathsf{invalid}})$, sets $\mathsf{ch} := (r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}})$, runs $(x^*, \mathsf{resp}) \xleftarrow{\$} \mathcal{P}^*_2(\mathsf{st}_{\mathcal{P}^*}, \mathsf{ch})$, and parses $(\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}}) \leftarrow \mathsf{resp}$. It outputs 1 if

$$\mathsf{ZAP.Verify}(r_{\mathsf{zap}}, (x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}), \pi_{\mathsf{zap}}) = \top$$

and either of the following is satisfied:

1. $\mathsf{dk}_{\mathsf{pke}} = \bot$, or
2. $\mathsf{dk}_{\mathsf{pke}} \neq \bot$ and $(x, \mathsf{PKE.Dec}(\mathsf{dk}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{pke}})) \notin \mathcal{R}$.

We can see that $\mathcal{A}$ perfectly simulates Game 2 (resp. Game 3) for $\mathcal{P}^*$ conditioned on that Invalid occurs if $\mathsf{ek}_{\mathsf{le}}$ is generated by $\mathsf{LE.LossyGen}(1^\kappa)$ (resp. $\mathsf{LE.InjGen}(1^\kappa)$), and $\mathcal{A}$ returns 1 if and only if Invalid occurs and $\mathcal{P}^*$ semi-wins in the corresponding game. Therefore, $\mathcal{A}$'s advantage to break the key indistinguishability of Π_{LE} is equal to $\left|\Pr[\mathsf{Win}'_3 \wedge \mathsf{Invalid}] - \Pr[\mathsf{Win}'_2 \wedge \mathsf{Invalid}]\right|$. Therefore, by the key indistinguishability of Π_{LE}, we have $\left|\Pr[\mathsf{Win}'_3 \wedge \mathsf{Invalid}] - \Pr[\mathsf{Win}'_2 \wedge \mathsf{Invalid}]\right| = \mathsf{negl}(\kappa)$. □

Lemma 4.4. *If Π_{IKC} satisfies invalid key certifiability, Π_{zap} satisfies adaptive statistical soundness, and Π_{HPG} is hard against non-uniform PPT adversaries, then we have $\Pr[\mathsf{Win}'_3] = \mathsf{negl}(\kappa)$.*

Proof. We divide the event Win'_3 into the following three cases.

$\mathsf{Win}'_{3.a}$**:** This is the event that Win'_3 occurs and we have $(x^*, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}) \notin \widetilde{\mathcal{L}}$.

$\mathsf{Win}'_{3.b}$**:** This is the event that Win'_3 occurs, we have $(x^*, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}) \in \widetilde{\mathcal{L}}$, and Invalid does not occur.

$\mathsf{Win}'_{3.c}$**:** This is the event that Win'_3 occurs, we have $(x^*, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}) \in \widetilde{\mathcal{L}}$, and Invalid occurs.

We show that none of them occurs with non-negligible probability.

Claim 4.5. If Π_{zap} satisfies adaptive statistical soundness, then we have $\Pr[\mathsf{Win}'_{3.a}] = \mathsf{negl}(\kappa)$.

Proof. We construct an unbounded-time adversary $\mathcal{A}$ against the adaptive soundness of Π_{zap} as follows:

$\mathcal{A}(r_{\mathsf{zap}})$**:** This algorithm runs $(\mathsf{pp} = (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}}), \mathsf{st}_{\mathcal{P}^*}) \xleftarrow{\$} \mathcal{P}^*_{\mathsf{Setup}}(1^\kappa)$, checks if Invalid occurs by a brute-force search, and finds $\mathsf{wit}_{\mathsf{invalid}}$ that satisfies $\mathsf{IKC.InvalidVerf}(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{wit}_{\mathsf{invalid}}) = \top$ if Invalid occurs. It generates $\mathsf{prob} \xleftarrow{\$} \mathsf{ProbGen}(1^\kappa)$, and generates $\mathsf{ek}_{\mathsf{le}}$ and π'_{zap} in either of the following ways:

1. If Invalid occurs, then it generates $\mathsf{ek}_{\mathsf{le}} \xleftarrow{\$} \mathsf{LE.InjGen}(1^\kappa)$ and $\pi'_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \mathsf{wit}_{\mathsf{invalid}}))$,
2. Otherwise, it generates $\mathsf{ek}_{\mathsf{le}} \xleftarrow{\$} \mathsf{LE.LossyGen}(1^\kappa; r_{\text{le-gen}})$and $\pi'_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), r_{\text{le-gen}}))$.

Then it sets $\mathsf{ch} := (r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}})$, runs $(x^*, \mathsf{resp}) \xleftarrow{\$} \mathcal{P}^*_2(\mathsf{st}_{\mathcal{P}^*}, \mathsf{ch})$, parses $(\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}}) \leftarrow \mathsf{resp}$, and outputs $(x^*, \pi_{\mathsf{zap}})$.

When $\mathsf{Win}'_{3,a}$ occurs, we have $\mathsf{ZAP.Verify}(r_{\mathsf{zap}}, (x^*, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}), \pi_{\mathsf{zap}}) = \top$ and $(x^*, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}) \notin \widetilde{\mathcal{L}}$. Therefore, by the soundness of Π_{zap}, we have $\Pr[\mathsf{Win}'_{3,a}] = \mathsf{negl}(\kappa)$. □

Claim 4.6. If Π_{IKC} satisfies invalid key certifiability and Π_{HPG} is hard against non-uniform PPT adversaries, then we have $\Pr[\mathsf{Win}'_{3.b}] = \mathsf{negl}(\kappa)$.

Proof. First, we remark that when $\mathsf{Win}'_{3.b}$ happens, we have $(x^*, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}) \in \widetilde{\mathcal{L}}_2$ since the requirement of that $\mathcal{P}^*$ semi-wins (and the correctness of Π_{IKC}) immediately imply that $(x^*, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}) \notin \widetilde{\mathcal{L}}_1$. Assuming $\Pr[\mathsf{Win}'_{3.b}]$ is non-negligible, we construct a non-uniform PPT algorithm $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ that breaks the hardness of Π_{HPG} as follows:

$\mathcal{A}_0(1^\kappa)$**:** This is an unbounded-time pre-computation phase of $\mathcal{A}$ that works as follows: It runs $(\mathsf{pp} = (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}}), \mathsf{st}_{\mathcal{P}^*}) \xleftarrow{\$} \mathcal{P}^*_{\mathsf{Setup}}(1^\kappa)$ and checks if Invalid occurs by a brute-force search, and immediately aborts if Invalid occurs. Otherwise, it finds a corresponding decryption key $\mathsf{dk}_{\mathsf{ikc}}$ to $\mathsf{ek}_{\mathsf{ikc}}$ by a brute-force search. (Note that such $\mathsf{dk}_{\mathsf{ikc}}$ exists when Invalid does not occur by the second requirement of invalid key certifiability.) Then it outputs $\mathsf{st}_{\mathcal{A}} := (\mathsf{st}_{\mathcal{P}^*}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{dk}_{\mathsf{ikc}}, r'_{\mathsf{zap}})$.

$\mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, \mathsf{prob})$**:** This is a PPT online phase of $\mathcal{A}$ that works as follows: It parses $(\mathsf{st}_{\mathcal{P}^*}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{dk}_{\mathsf{ikc}}, r'_{\mathsf{zap}}) \leftarrow \mathsf{st}_{\mathcal{A}}$, generates $r_{\mathsf{zap}} \xleftarrow{\$} \{0,1\}^\ell$, $\mathsf{ek}_{\mathsf{le}} := \mathsf{LE.LossyGen}(1^\kappa; r_{\text{le-gen}})$, and $\pi'_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), r_{\text{le-gen}})$, sets $\mathsf{ch} := (r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}})$, runs $(x^*, \mathsf{resp}) \xleftarrow{\$} \mathcal{P}^*_2(\mathsf{st}_{\mathcal{P}^*}, \mathsf{ch})$, parses $(\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}}) \leftarrow \mathsf{resp}$, and outputs $\mathsf{IKC.Dec}(\mathsf{dk}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{ikc}})$.

It is easy to see that $\mathcal{A}$ perfectly simulates the environment of Game 3 for $\mathcal{P}^*$ conditioned on that Invalid does not occur. Suppose that $\mathsf{Win}'_{3.b}$ happens. Then $\mathcal{A}$ succeeds in finding $\mathsf{dk}_{\mathsf{ikc}}$ corresponding to $\mathsf{ek}_{\mathsf{ikc}}$ since the condition for $\mathsf{Win}'_{3.b}$ includes that Invalid does not occur. Moreover, as discussed above, we have $(x^*, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}) \in \widetilde{\mathcal{L}}_2$, which means that there exists $(\mathsf{sol}, r_{\mathsf{ikc\text{-}enc}}, r_{\mathsf{le\text{-}enc}})$ such that

$$\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol}) = \top,$$

$$\mathsf{ct}_{\mathsf{ikc}} = \mathsf{IKC.Enc}(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{sol}; r_{\mathsf{ikc\text{-}enc}}),$$

$$\mathsf{ct}_{\mathsf{le}} = \mathsf{LE.Enc}(\mathsf{ek}_{\mathsf{le}}, \mathsf{sol}; r_{\mathsf{le\text{-}enc}}).$$

When this happens, it is clear that $\mathcal{A}$ succeeds in outputting a valid witness sol such that $\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol}) = \top$. Therefore $\Pr[\mathsf{Win}'_{3.b}] = \mathsf{negl}(\kappa)$ as long as Π_{HPG} is hard against non-uniform PPT adversaries. □

Claim 4.7. If Π_{HPG} is hard against non-uniform PPT adversaries, then we have $\Pr[\mathsf{Win}'_{3.c}] = \mathsf{negl}(\kappa)$.

Proof. The proof of this claim is very similar to that of Claim 4.6 except that $\mathcal{A}$ extracts the solution of Π_{HPG} from $\mathsf{ct}_{\mathsf{le}}$ instead of $\mathsf{ct}_{\mathsf{ikc}}$. Similarly to the proof of Claim 4.6, when $\mathsf{Win}'_{3.c}$ happens, we have $(x^*, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}) \in \widetilde{\mathcal{L}}_2$. Assuming $\Pr[\mathsf{Win}'_{3.c}]$ is non-negligible, we construct a non-uniform classical polynomial-time algorithm $\mathcal{A}$ that breaks Π_{HPG} as follows:

$\mathcal{A}_0(1^\kappa)$**:** This is an unbounded-time pre-computation phase of $\mathcal{A}$ that works as follows: It runs $(\mathsf{pp} = (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}}), \mathsf{st}_{\mathcal{P}^*}) \xleftarrow{\$} \mathcal{P}^*_{\mathsf{Setup}}(1^\kappa)$ and checks if Invalid occurs by a brute-force search, and immediately aborts if Invalid does not occur. Otherwise, it finds $\mathsf{wit}_{\mathsf{invalid}}$ such that $\mathsf{IKC.InvalidVerf}(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{wit}_{\mathsf{invalid}}) = \top$ and outputs $\mathsf{st}_{\mathcal{A}} := (\mathsf{st}_{\mathcal{P}^*}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{wit}_{\mathsf{invalid}}, r'_{\mathsf{zap}})$.

$\mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, \mathsf{prob})$**:** This is a PPT online phase of $\mathcal{A}$ that works as follows: It parses $(\mathsf{st}_{\mathcal{P}^*}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{wit}_{\mathsf{invalid}}, r'_{\mathsf{zap}}) \leftarrow \mathsf{st}_{\mathcal{A}}$, generates $r_{\mathsf{zap}} \xleftarrow{\$} \{0,1\}^\ell$, $(\mathsf{ek}_{\mathsf{le}}, \mathsf{dk}_{\mathsf{le}}) \xleftarrow{\$} \mathsf{LE.InjGen}(1^\kappa)$, and $\pi'_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \mathsf{wit}_{\mathsf{invalid}})$, sets $\mathsf{ch} := (r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}})$, runs $(x^*, \mathsf{resp}) \xleftarrow{\$} \mathcal{P}^*_2(\mathsf{st}_{\mathcal{P}^*}, \mathsf{ch})$, parses $(\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}}) \leftarrow \mathsf{resp}$, and outputs $\mathsf{LE.Dec}(\mathsf{dk}_{\mathsf{le}}, \mathsf{ct}_{\mathsf{ikc}})$.

It is easy to see that $\mathcal{A}$ perfectly simulates the environment of Game 3 for $\mathcal{P}^*$ conditioned on that Invalid occurs. As discussed above, when $\mathsf{Win}'_{3.c}$ occurs, we have $(x^*, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}) \in \widetilde{\mathcal{L}}_2$ which means that there exists $(\mathsf{sol}, r_{\mathsf{ikc\text{-}enc}}, r_{\mathsf{le\text{-}enc}})$ such that

$$\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol}) = \top,$$

$$\mathsf{ct}_{\mathsf{ikc}} = \mathsf{IKC.Enc}(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{sol}; r_{\mathsf{ikc\text{-}enc}}),$$

$$\mathsf{ct}_{\mathsf{le}} = \mathsf{LE.Enc}(\mathsf{ek}_{\mathsf{le}}, \mathsf{sol}; r_{\mathsf{le\text{-}enc}}).$$

When this happens, it is clear that $\mathcal{A}$ succeeds in outputting a valid witness sol such that $\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol}) = \top$ by the correctness of Π_{LE} in the injective mode. Therefore $\Pr[\mathsf{Win}'_{3.c}] = \mathsf{negl}(\kappa)$ as long as Π_{HPG} is hard against non-uniform PPT adversaries. □

Combining Claims 4.5 to 4.7, we have $\Pr[\mathsf{Win}'_3] = \mathsf{negl}(\kappa)$, which concludes the proof of Lemma 4.4. □

Combining Lemmata 4.2 to 4.4, we have $\Pr[\mathsf{Win}'_1] = \mathsf{negl}(\kappa)$, which concludes the proof of Theorem 4.1. □

Reusable Quantum-Simulation Zero-Knowledge.

Theorem 4.8. *If Π_{PKE} satisfies CPA security against QPT adversaries, Π_{IKC} satisfies invalid key certifiability and CPA security against QPT adversaries, Π_{LE} satisfies lossiness of lossy keys, Π_{HPG} satisfies quantum solvability, Π_{ZAP} satisfies adaptive computational witness indistinguishability against non-uniform PPT adversaries, and Π'_{ZAP} satisfies adaptive statistical soundness, then Π_{ZK} satisfies the reusable quantum-simulation zero-knowledge against PPT adversaries.*

Proof. We first describe a QPT simulator $\mathcal{S}$.

$\mathcal{S}(\mathsf{pp}, \mathsf{ch}, x, 1^{|w|})$**:** $\mathcal{S}$ is given $\mathsf{pp} = (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}})$, $\mathsf{ch} = (r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}})$, a statement x, and a witness length $1^{|w|}$ as input, and works as follows.

1. Return $\bot$ if $\mathsf{VerProb}(1^\kappa, \mathsf{prob}) = \bot$ or $\mathsf{ZAP.Verify}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \pi'_{\mathsf{zap}}) = \bot$.
2. Generate $\mathsf{sol} \xleftarrow{\$} \mathsf{Solve}(\mathsf{prob})$ (by using a QPT computation). If $\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol}) = \bot$, immediately return $\bot$ and halt. Otherwise, generate $\mathsf{ct}_{\mathsf{ikc}} \xleftarrow{\$} \mathsf{Enc}(\mathsf{sol}; r_{\text{ikc-enc}})$ and $\mathsf{ct}_{\mathsf{le}} \xleftarrow{\$} \mathsf{LE.Enc}(\mathsf{ek}_{\mathsf{le}}, \mathsf{sol}; r_{\text{le-enc}})$.
3. Generate $\mathsf{ct}_{\mathsf{pke}} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{ek}_{\mathsf{pke}}, 0^{|w|})$.
4. Generate $\pi_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}(r_{\mathsf{zap}}, (x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}), (\mathsf{sol}, r_{\text{ikc-enc}}, r_{\text{le-enc}}))$.
5. Return $\mathsf{resp} = (\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}})$.

We prove that resp simulated by $\mathcal{S}$ is indistinguishable from the real one by considering the following sequence of games between a PPT adversary $\mathcal{A}$ and a challenger.

Game 1**:** This is the real experiment. That is, the challenger generates $(\mathsf{ek}_{\mathsf{pke}}, \mathsf{dk}_{\mathsf{pke}}) := \mathsf{PKE.KeyGen}(1^\kappa; r_{\text{pke-gen}})$, $(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{dk}_{\mathsf{ikc}}) \xleftarrow{\$} \mathsf{IKC.KeyGen}(1^\kappa)$, $r'_{\mathsf{zap}} \xleftarrow{\$} \{0,1\}^\ell$, and sets $\mathsf{pp} := (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}})$ and $\mathsf{sp} := (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}}, \mathsf{dk}_{\mathsf{pke}}, r_{\text{pke-gen}})$. $\mathcal{A}$ is given pp as input and access to an oracle $\mathcal{O}_{\mathsf{real}}(\mathsf{ch}, x, w)$ that works as follows:

$\mathcal{O}_{\mathsf{real}}(\mathsf{ch}, x, w)$**:** It immediately returns $\bot$ if $(x, w) \notin \mathcal{R}$. Otherwise, it parses $(r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}}) \leftarrow \mathsf{ch}$ and works as follows:

1. Return $\bot$ if $\mathsf{VerProb}(1^\kappa, \mathsf{prob}) = \bot$, or $\mathsf{ZAP.Verify}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \pi'_{\mathsf{zap}}) = \bot$.
2. Generate $\mathsf{ct}_{\mathsf{ikc}} \xleftarrow{\$} \mathsf{IKC.Enc}(\mathsf{ek}_{\mathsf{ikc}}, 0^{|\mathsf{sol}|})$ and $\mathsf{ct}_{\mathsf{le}} \xleftarrow{\$} \mathsf{LE.Enc}(\mathsf{ek}_{\mathsf{le}}, 0^{|\mathsf{sol}|})$.
3. Generate $\mathsf{ct}_{\mathsf{pke}} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{ek}_{\mathsf{pke}}, w; r_{\text{pke-enc}})$.
4. Generate $\pi_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}(r_{\mathsf{zap}}, (x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}), (w, \mathsf{dk}_{\mathsf{pke}}, r_{\text{pke-gen}}, r_{\text{pke-enc}}))$.
5. Return $\mathsf{resp} := (\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}})$.

$\mathcal{A}$'s final output is considered as the output of this game.

Game 2**:** This game is identical to the previous game except that the oracle $\mathcal{O}_{\mathsf{real}}(\mathsf{ch}, x, w)$ is replaced with $\mathcal{O}_{\mathsf{hyb}}(\mathsf{ch}, x, w)$ that works similarly to $\mathcal{O}_{\mathsf{real}}(\mathsf{ch}, x, w)$ except that Step 2 and 4 are replaced with the corresponding steps of $\mathcal{S}$. That is, $\mathcal{O}_{\mathsf{hyb}}(\mathsf{ch}, x, w)$ works as follows:

$\mathcal{O}_{\mathsf{hyb}}(\mathsf{ch}, x, w)$**:** It immediately returns $\bot$ if $(x, w) \notin \mathcal{R}$. Otherwise, it parses $(r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}}) \leftarrow \mathsf{ch}$ and works as follows:

1. Return $\bot$ if $\mathsf{VerProb}(1^\kappa, \mathsf{prob}) = \bot$, or $\mathsf{ZAP.Verify}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \pi'_{\mathsf{zap}}) = \bot$.
2. Generate $\mathsf{sol} \xleftarrow{\$} \mathsf{Solve}(\mathsf{prob})$. If $\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol}) = \bot$, immediately return $\bot$. Otherwise, generate $\mathsf{ct}_{\mathsf{ikc}} \xleftarrow{\$} \mathsf{Enc}(\mathsf{sol}; r_{\text{ikc-enc}})$ and $\mathsf{ct}_{\mathsf{le}} \xleftarrow{\$} \mathsf{LE.Enc}(\mathsf{ek}_{\mathsf{le}}, \mathsf{sol}; r_{\text{le-enc}})$.
3. Generate $\mathsf{ct}_{\mathsf{pke}} := \mathsf{PKE.Enc}(\mathsf{ek}_{\mathsf{pke}}, w; r_{\text{pke-enc}})$.
4. Generate $\pi_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}(r_{\mathsf{zap}}, (x, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}), (\mathsf{sol}, r_{\text{ikc-enc}}, r_{\text{le-enc}}))$.
5. Return $\mathsf{resp} := (\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}})$.

Game 3**:** This game is identical to the previous game except that the oracle $\mathcal{O}_{\mathsf{real}}(\mathsf{ch}, x, w)$ is replaced with $\mathcal{O}_{\mathsf{sim}}(\mathsf{ch}, x, w)$ that works as follows:

$\mathcal{O}_{\mathsf{sim}}(\mathsf{ch}, x, w)$**:** It immediately returns $\bot$ if $(x, w) \notin \mathcal{R}$. Otherwise, it outputs $\mathcal{S}(\mathsf{pp}, \mathsf{ch}, x)$.

Let E_i be the event that Game i returns 1. What we have to prove is that we have $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_1]| = \mathsf{negl}(\kappa)$. For proving this, we prove the following lemmata.

Lemma 4.9. *If Π_{IKC} satisfies invalid key certifiability and CPA security against QPT adversaries, Π_{LE} satisfies lossiness of lossy keys, Π_{HPG} satisfies quantum solvability, Π_{ZAP} satisfies adaptive computational witness indistinguishability against non-uniform PPT adversaries, and Π'_{ZAP} satisfies adaptive statistical soundness, then we have $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_1]| = \mathsf{negl}(\kappa)$.*

Proof. The differences between Game 2 and Game 1 are as follows:

1. $\mathsf{ct}_{\mathsf{ikc}}$ is an encryption of sol instead of $0^{|\mathsf{sol}|}$, and
2. $\mathsf{ct}_{\mathsf{le}}$ is an encryption of sol instead of $0^{|\mathsf{sol}|}$, and
3. π_{zap} is generated by using a witness of $\widetilde{\mathcal{L}}_2$ instead of $\widetilde{\mathcal{L}}_1$.

Roughly, the first difference is indistinguishable by the CPA security of Π_{IKC} against QPT adversaries. The second difference is indistinguishable due to the following reasons. (1) If $\mathsf{ek}_{\mathsf{le}}$ is a lossy key, encryptions of sol and $0^{|\mathsf{sol}|}$ are statistically indistinguishable. (2) If $\mathsf{ek}_{\mathsf{le}}$ is not a lossy key, we have $\mathsf{ZAP.Verify}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \pi'_{\mathsf{zap}}) = \bot$ with overwhelming probability by the soundness of Π'_{ZAP} noting that $\mathsf{ek}_{\mathsf{ikc}}$ is honestly generated. In this case, $\mathsf{ct}_{\mathsf{le}}$ is not given to the adversary. The third difference is indistinguishable by the witness indistinguishability of Π_{ZAP}. We would be able to turn this intuition into a formal proof in a straightforward manner if we assumed witness indistinguishability against *quantum* adversaries. However, since we only assume witness indistinguishability against non-uniform *classical* adversaries, we have to be careful about the order of game hops.[18] Namely, if we first make the modifications 1 and 2 for all queries, then we cannot make the modification 3 since the game involves quantum computations in every query. To circumvent this issue, we make the above modifications for each query one-by-one similarly to [GRS+11]. In this way, we can ensure that all quantum computations can be done in the pre-computation stage when making the modification 3 for each query, and the proof goes through even with witness indistinguishability against non-uniform PPT adversaries. The detail is described below:

Let q be the number of $\mathcal{A}$'s queries.[19] For each $(i, j) \in ([q] \times [4]) \cup \{(q+1, 1)\}$, we consider further hybrids Game $1.i.j$ between Game 1 and Game 2 as follows: (In the descriptions of these games, main differences from the previous game is highlighted by red underlines.)

Game $1.i.1$**:** This game is identical to Game 1 except that the oracle $\mathcal{O}_{\mathsf{real}}$ is replaced with $\mathcal{O}_{1.i.1}$ that works similarly to $\mathcal{O}_{\mathsf{hyb}}$ for the first $i-1$ queries and similarly to $\mathcal{O}_{\mathsf{real}}$ for the rest of queries.

Game $1.i.2$**:** This game is identical to Game 1 except that the oracle $\mathcal{O}_{\mathsf{real}}$ is replaced with $\mathcal{O}_{1.i.2}$ that works similarly to $\mathcal{O}_{\mathsf{hyb}}$ for the first $i-1$ queries, similarly to $\mathcal{O}_{\mathsf{real}}$ for the last $q-i$ queries, and works as follows for the i-th query: Let $(\mathsf{ch} = (r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}}), x, w)$ be $\mathcal{A}$'s i-th query. Then $\mathcal{O}_{1.i.2}$ works as follows:

1. Return $\bot$ if $\mathsf{VerProb}(1^\kappa, \mathsf{prob}) = \bot$, or $\mathsf{ZAP.Verify}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \pi'_{\mathsf{zap}}) = \bot$.
2. Generate $\mathsf{sol} \xleftarrow{\$} \mathsf{Solve}(\mathsf{prob})$. If $\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol}) = \bot$, immediately return $\bot$. Otherwise, generate $\mathsf{ct}_{\mathsf{ikc}} \xleftarrow{\$} \mathsf{IKC.Enc}(\mathsf{ek}_{\mathsf{ikc}}, 0^{|\mathsf{sol}|})$ and $\mathsf{ct}_{\mathsf{le}} \xleftarrow{\$} \mathsf{LE.Enc}(\mathsf{ek}_{\mathsf{le}}, 0^{|\mathsf{sol}|})$.
3. Generate $\mathsf{ct}_{\mathsf{pke}} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{ek}_{\mathsf{pke}}, w; r_{\mathsf{pke\text{-}enc}})$.
4. Generate $\pi_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}(r_{\mathsf{zap}}, (x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}), (w, \mathsf{dk}_{\mathsf{pke}}, r_{\mathsf{pke\text{-}gen}}, r_{\mathsf{pke\text{-}enc}}))$.
5. Return $\mathsf{resp} := (\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}})$.

Game $1.i.3$**:** This game is identical to Game 1 except that the oracle $\mathcal{O}_{\mathsf{real}}$ is replaced with $\mathcal{O}_{1.i.3}$ that works similarly to $\mathcal{O}_{\mathsf{hyb}}$ for the first $i-1$ queries, similarly to $\mathcal{O}_{\mathsf{real}}$ for the last $q-i$ queries, and works as follows for the i-th query: Let $(\mathsf{ch} = (r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}}), x, w)$ be $\mathcal{A}$'s i-th query. Then $\mathcal{O}_{1.i.3}$ works as follows:

1. Return $\bot$ if $\mathsf{VerProb}(1^\kappa, \mathsf{prob}) = \bot$, or $\mathsf{ZAP.Verify}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \pi'_{\mathsf{zap}}) = \bot$.

[18] Note that there is no known ZAP with witness indistinguishability against QPT adversaries based on the (quantum) polynomial hardness of standard assumptions.

[19] We can assume that the number of $\mathcal{A}$'s queries is a fixed number that only depends on the security parameter without loss of generality.

2. Generate $\mathsf{sol} \xleftarrow{\$} \mathsf{Solve}(\mathsf{prob})$. If $\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol}) = \bot$, immediately return $\bot$. Otherwise, generate $\mathsf{ct}_{\mathsf{ikc}} \xleftarrow{\$} \mathsf{IKC.Enc}(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{sol}; r_{\mathsf{ikc\text{-}enc}})$ and $\mathsf{ct}_{\mathsf{le}} \xleftarrow{\$} \mathsf{LE.Enc}(\mathsf{ek}_{\mathsf{le}}, 0^{|\mathsf{sol}|})$.
3. Generate $\mathsf{ct}_{\mathsf{pke}} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{ek}_{\mathsf{pke}}, w; r_{\mathsf{pke\text{-}enc}})$.
4. Generate $\pi_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}(r_{\mathsf{zap}}, (x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}), (w, \mathsf{dk}_{\mathsf{pke}}, r_{\mathsf{pke\text{-}gen}}, r_{\mathsf{pke\text{-}enc}}))$.
5. Return $\mathsf{resp} := (\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}})$.

Game $1.i.4$**:** This game is identical to Game 1 except that the oracle $\mathcal{O}_{\mathsf{real}}$ is replaced with $\mathcal{O}_{1.i.4}$ that works similarly to $\mathcal{O}_{\mathsf{hyb}}$ for the first $i-1$ queries, similarly to $\mathcal{O}_{\mathsf{real}}$ for the last $q-i$ queries, and works as follows for the i-th query: Let $(\mathsf{ch} = (r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}}), x, w)$ be $\mathcal{A}$'s i-th query. Then $\mathcal{O}_{1.i.4}$ works as follows:

1. Return $\bot$ if $\mathsf{VerProb}(1^\kappa, \mathsf{prob}) = \bot$, or $\mathsf{ZAP.Verify}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \pi'_{\mathsf{zap}}) = \bot$.
2. Generate $\mathsf{sol} \xleftarrow{\$} \mathsf{Solve}(\mathsf{prob})$. If $\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol}) = \bot$, immediately return $\bot$. Otherwise, generate $\mathsf{ct}_{\mathsf{ikc}} \xleftarrow{\$} \mathsf{IKC.Enc}(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{sol}; r_{\mathsf{ikc\text{-}enc}})$ and $\mathsf{ct}_{\mathsf{le}} \xleftarrow{\$} \mathsf{LE.Enc}(\mathsf{ek}_{\mathsf{le}}, \mathsf{sol}; r_{\mathsf{le\text{-}enc}})$.
3. Generate $\mathsf{ct}_{\mathsf{pke}} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{ek}_{\mathsf{pke}}, w; r_{\mathsf{pke\text{-}enc}})$.
4. Generate $\pi_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}(r_{\mathsf{zap}}, (x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}), (w, \mathsf{dk}_{\mathsf{pke}}, r_{\mathsf{pke\text{-}gen}}, r_{\mathsf{pke\text{-}enc}}))$.
5. Return $\mathsf{resp} := (\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}})$.

Clearly, Game $1.i.1$ is identical to Game 1 and Game $1.(q+1).1$ is identical to Game 2. Let $\mathsf{E}_{1,i,j}$ be the event that Game $1.i.j$ returns 1. We prove the following claims:

Claim 4.10. If Π_{HPG} satisfies quantum solvability, then we have $|\Pr[\mathsf{E}_{1.i.2}] - \Pr[\mathsf{E}_{1.i.1}]| = \mathsf{negl}(\kappa)$ for any $i \in [q]$.

Proof. Game $1.i.2$ is identical to Game $1.i.1$ except that when responding to $\mathcal{A}$'s i-th query, the oracle generates $\mathsf{sol} \xleftarrow{\$} \mathsf{Solve}(\mathsf{prob})$, and returns $\bot$ if $\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol}) = \bot$. By the quantum solvability of Π_{HPG}, this happens with negligible probability. Therefore, Claim 4.10 follows. □

Claim 4.11. If Π_{IKC} satisfies CPA security against QPT adversaries, then we have $|\Pr[\mathsf{E}_{1.i.3}] - \Pr[\mathsf{E}_{1.i.2}]| = \mathsf{negl}(\kappa)$ for any $i \in [q]$.

Proof. Game $1.i.3$ is identical to Game $1.i.2$ except that $\mathsf{ct}_{\mathsf{ikc}}$ is an encryption of sol instead of $0^{|\mathsf{sol}|}$ in a response to i-th query. We observe that the public key $\mathsf{ek}_{\mathsf{ikc}}$ used for generating $\mathsf{ct}_{\mathsf{ikc}}$ is honestly generated, and the corresponding decryption key $\mathsf{dk}_{\mathsf{ikc}}$ is not used at all in these games. Therefore, it is straightforward to reduce the indistinguishability between these games to CPA security of Π_{IKC} against QPT adversaries. Note that we need CPA security against QPT adversaries (instead of non-uniformly classical adversaries) since these games use QPT computation for running Solve *after generating* $\mathsf{ek}_{\mathsf{ikc}}$. □

Claim 4.12. If Π_{IKC} satisfies invalid key certifiability, Π_{LE} satisfies lossiness of lossy keys and Π'_{ZAP} satisfies adaptive statistical soundness, then we have $|\Pr[\mathsf{E}_{1.i.4}] - \Pr[\mathsf{E}_{1.i.3}]| = \mathsf{negl}(\kappa)$ for any $i \in [q]$.

Proof. Game $1.i.4$ is identical to Game $1.i.3$ except that $\mathsf{ct}_{\mathsf{le}}$ is an encryption of sol instead of $0^{|\mathsf{sol}|}$ in a response to i-th query. The distributions of them are statistically indistinguishable if the corresponding encryption key $\mathsf{ek}_{\mathsf{le}}$ is in the lossy mode (i.e., generated by $\mathsf{LE.LossyGen}(1^\kappa)$). Moreover, $\mathsf{ct}_{\mathsf{le}}$ is given to $\mathcal{A}$ only when $\mathsf{ZAP.Verify}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \pi'_{\mathsf{zap}}) = \top$. Therefore, (even an unbounded-time) adversary may distinguish these two games with non-negligible advantage only if an event Bad that $\mathsf{ZAP.Verify}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \pi'_{\mathsf{zap}}) = \top$ and $\mathsf{ek}_{\mathsf{le}}$ is not in the lossy mode occurs with non-negligible probability. Thus, it suffices to prove that we have $\Pr[\mathsf{Bad}] = \mathsf{negl}(\kappa)$. By the definition of $\widetilde{\mathcal{L}}'_1$, we have $(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}) \notin \widetilde{\mathcal{L}}'_1$ for any $\mathsf{ek}_{\mathsf{ikc}}$ if $\mathsf{ek}_{\mathsf{le}}$ is not in the lossy mode. Moreover, we observe that since $\mathsf{ek}_{\mathsf{ikc}}$ is honestly generated in these games, the first requirement of the invalid key certifiability ensures that there does not exist $\mathsf{wit}_{\mathsf{invalid}}$ such that $\mathsf{IKC.InvalidVerf}(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{wit}_{\mathsf{invalid}}) = \top$, which implies $(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}) \notin \widetilde{\mathcal{L}}'_2$ for any $\mathsf{ek}_{\mathsf{le}}$. Combining them, if $\mathsf{ek}_{\mathsf{le}}$ is not in the lossy mode, $(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}) \notin \widetilde{\mathcal{L}}'$. On the other hand, the adaptive statistical soundness of Π'_{ZAP} ensures that an event that $\mathsf{ZAP.Verify}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \pi'_{\mathsf{zap}}) = \top$ and $(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}) \notin \widetilde{\mathcal{L}}'$ occurs with negligible probability. This implies $\Pr[\mathsf{Bad}] = \mathsf{negl}(\kappa)$, which completes the proof of Claim 4.12. □

Claim 4.13. If Π_{ZAP} satisfies adaptive computational witness indistinguishability against non-uniform PPT adversaries, then we have $\left|\Pr[\mathsf{E}_{1.(i+1).1}] - \Pr[\mathsf{E}_{1.i.4}]\right| = \mathsf{negl}(\kappa)$ for any $i \in [q]$.

Proof. Game $1.(i+1).1$ is identical to Game $1.i.4$ except that π_{zap} is generated by using a witness of $\widetilde{\mathcal{L}}_2$ instead of $\widetilde{\mathcal{L}}_1$ in a response to i-th query. Noting that all quantum computations in these games are done *before* π_{zap} in a response to i-th query is generated, we can consider these quantum computations as part of pre-computation, and thus we can reduce the indistinguishability between these two games to the witness indistinguishability of Π_{ZAP} against non-uniform PPT adversaries (instead of quantum adversaries). For completeness, we give the full description of the reduction algorithm below.

We construct a non-uniform PPT adversary $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ against the witness indistinguishability of Π_{ZAP} as follows:

$\mathcal{B}_0(1^\kappa)$**:** This is an unbounded-time pre-computation phase of $\mathcal{B}$ that works as follows:[20] It generates $(\mathsf{ek}_{\mathsf{pke}}, \mathsf{dk}_{\mathsf{pke}}) := \mathsf{PKE.KeyGen}(1^\kappa; r_{\mathsf{pke\text{-}gen}})$, $(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{dk}_{\mathsf{ikc}}) \xleftarrow{\$} \mathsf{IKC.KeyGen}(1^\kappa)$, $r'_{\mathsf{zap}} \xleftarrow{\$} \{0,1\}^\ell$, and sets $\mathsf{pp} := (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}})$ and $\mathsf{sp} := (\mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, r'_{\mathsf{zap}}, \mathsf{dk}_{\mathsf{pke}}, r_{\mathsf{pke\text{-}gen}})$. Then it runs $\mathcal{A}(\mathsf{pp})$ until it makes i-th query where $\mathcal{B}_0$ responds to the first $i-1$ queries by using $\mathcal{O}_{\mathsf{hyb}}$ as described in Game 2. Let $(\mathsf{ch} = (r_{\mathsf{zap}}, \mathsf{prob}, \mathsf{ek}_{\mathsf{le}}, \pi'_{\mathsf{zap}}), x, w)$ be $\mathcal{A}$'s i-th query and $\mathsf{st}_{\mathcal{A}}$ be the snapshot of $\mathcal{A}$ immediately after making the i-th query. Then $\mathcal{B}_0$ generates $\mathsf{sol} \xleftarrow{\$} \mathsf{Solve}(\mathsf{prob})$, $\mathsf{ct}_{\mathsf{ikc}} \xleftarrow{\$} \mathsf{IKC.Enc}(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{sol}; r_{\mathsf{ikc\text{-}enc}})$, $\mathsf{ct}_{\mathsf{le}} \xleftarrow{\$} \mathsf{LE.Enc}(\mathsf{ek}_{\mathsf{le}}, \mathsf{sol}; r_{\mathsf{le\text{-}enc}})$, and $\mathsf{ct}_{\mathsf{pke}} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{ek}_{\mathsf{pke}}, w; r_{\mathsf{pke\text{-}enc}})$, and sets $x' := (x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}})$. It sets $\mathsf{flag} := \bot$ and $w'_0 = w'_1 := (w, \mathsf{dk}_{\mathsf{pke}}, r_{\mathsf{pke\text{-}gen}}, r_{\mathsf{pke\text{-}enc}})$ if $\mathsf{VerProb}(1^\kappa, \mathsf{prob}) = \bot$, $\mathsf{ZAP.Verify}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \pi'_{\mathsf{zap}}) = \bot$, or $\mathsf{VerSol}(\mathsf{prob}, \mathsf{sol}) = \bot$. Otherwise, it sets $\mathsf{flag} := \top$, $w'_0 := (w, \mathsf{dk}_{\mathsf{pke}}, r_{\mathsf{pke\text{-}gen}}, r_{\mathsf{pke\text{-}enc}})$, and $w'_1 := (\mathsf{sol}, r_{\mathsf{ikc\text{-}enc}}, r_{\mathsf{le\text{-}enc}})$. Finally, it outputs flag, x', w'_0, w'_1, and $\mathsf{st}_{\mathcal{B}} := (\mathsf{flag}, \mathsf{st}_{\mathcal{A}}, \mathsf{sp}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}})$.

$\mathcal{B}_1(\mathsf{st}_{\mathcal{B}}, \pi_{\mathsf{zap}})$**:** This is a PPT online phase of $\mathcal{B}$ that works as follows: It parses $(\mathsf{flag}, \mathsf{st}_{\mathcal{A}}, \mathsf{sp}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}) \leftarrow \mathsf{st}_{\mathcal{B}}$. It sets $\mathsf{resp} := \bot$ if $\mathsf{flag} = \bot$ and otherwise $\mathsf{resp} := (\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}})$. $\mathcal{B}_1$ restarts $\mathcal{A}$ from the point when it has just made its i-th query by using the snapshot $\mathsf{st}_{\mathcal{A}}$, returns resp as a response from the oracle to the i-th query, and runs the rest of execution of $\mathcal{A}$ by simulating an oracle by $\mathcal{O}_{\mathsf{real}}$ as described in Game 1. Finally, $\mathcal{B}_1$ outputs a bit output by $\mathcal{A}$.

We can see that $\mathcal{B}$ perfectly simulates Game $1.(i+1).1$ (resp. Game $1.i.4$) for $\mathcal{A}$ if π_{zap} is generated by using the witness w'_1 (resp. w'_0). Moreover, for any (x', w'_0, w'_1) output by $\mathcal{B}_0$, both w'_0 and w'_1 are valid witnesses for $x' \in \widetilde{\mathcal{L}}$. Therefore, $\mathcal{B}$'s advantage to break the witness indistinguishability of Π_{ZAP} is equal to $\left|\Pr[\mathsf{E}_{1.(i+1).1}] - \Pr[\mathsf{E}_{1.i.4}]\right|$. Therefore, this is negligible by the witness indistinguishability of Π_{ZAP}. □

By combining Claims 4.10 to 4.13, we have $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_1]| = \left|\Pr[\mathsf{E}_{1.(q+1).1}] - \Pr[\mathsf{E}_{1.1.1}]\right| = \mathsf{negl}(\kappa)$, which completes the proof of Lemma Lemma 4.9. □

Lemma 4.14. *Π_{PKE} satisfies CPA security against QPT adversaries, then we have* $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\kappa)$.

Proof. Game 3 is identical to Game 2 except that $\mathsf{ct}_{\mathsf{pke}}$ is an encryption of $0^{|w|}$ instead of w in a response to each query. Thanks to the modification made in Game 2, $\mathsf{dk}_{\mathsf{pke}}$ and $r_{\mathsf{pke\text{-}gen}}$ are not used at all in this game. Therefore, by a straightforward hybrid argument, we can reduce the indistinguishability between these games to CPA security of Π_{PKE} against QPT adversaries. Note that we need CPA security against QPT adversaries (instead of non-uniformly classical adversaries) since these games use QPT computation for running Solve *after generating* $\mathsf{ek}_{\mathsf{pke}}$. □

Combining Lemmata 4.9 and 4.14, we have $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_1]| = \mathsf{negl}(\kappa)$, which completes the proof of Theorem 4.8. □

4.4 Instantiations

As mentioned in Sections 2 and 3, we can instantiate Π_{PKE} and Π_{IKC} under the QPT hardness of LWE, Π_{LE} under the non-uniform PPT hardness of DDH, DLIN, QR, or DCR, Π_{HPG} under the non-uniform PPT hardness of factoring or DL, and Π_{zap} under the non-uniform PPT hardness of factoring, SXDH in pairing groups, or DLIN in pairing groups.

[20] Actually, this phase runs in QPT.

Noting that the QR or DCR assumption implies the factoring assumption, the SXDH assumption implies the DDH assumption, and the SXDH or DLIN assumption implies the DL assumption, we can instantiate all the building blocks if we assume the QPT hardness of LWE and non-uniform PPT hardness of either of QR, DCR, SXDH in pairing groups, or DLIN in pairing groups.

5 Round-Optimal Blind Signatures

In this section, we construct round-optimal blind signatures.

5.1 Construction

Building blocks. We construct a round-optimal blind signature scheme based on the following building blocks.

- $\Pi_{\mathsf{Sig}} = (\mathsf{SigGen}, \mathsf{Sign}, \mathsf{SigVerify})$ is a digital signature scheme that is EUF-CMA against QPT adversaries. We assume that Sign is deterministic. This can be assumed without loss of generality by derandomizing the signing algorithm by using a quantumly secure PRF (which is only required to be secure against QPT adversaries that just make classical queries).
- $\Pi_{\mathsf{SFE}} = (\mathsf{Receiver}, \mathsf{Sender}, \mathsf{Derive})$ is an SFE protocol that satisfies receiver's security against non-uniform PPT adversaries and quantum-extraction sender's security against QPT adversaries.
- Com is a perfectly-binding non-interactive commitment with computational hiding against QPT adversaries.
- $\Pi_{\mathsf{ZK}} = (\mathsf{Setup}, \mathcal{V}_1, \mathcal{P}_2, \mathcal{V}_{\mathsf{out}})$ is blind-signature-conforming zero-knowledge arguments for a language $\mathcal{L}$, which is defined as follows: We have $(\mathsf{com}, \mathsf{sfe}_1, \mathsf{sfe}_2) \in \mathcal{L}$ if there exists $(\mathsf{ssk}, r_{\mathsf{com}}, r_{\mathsf{sfe}})$ such that

$$\mathsf{com} = \mathsf{Com}(\mathsf{ssk}; r_{\mathsf{com}})$$

$$\mathsf{sfe}_2 = \mathsf{Sender}(1^\kappa, \mathsf{sfe}_1, \mathsf{Sign}(\mathsf{ssk}, \cdot); r_{\mathsf{sfe}})$$

Construction. Our construction of a round-optimal blind signature scheme $\Pi_{\mathsf{BS}} = (\mathsf{BSGen}, \mathcal{U}_1, \mathcal{S}_2, \mathcal{U}_{\mathsf{der}}, \mathsf{BSVerify})$ is described as follows.

$\mathsf{BSGen}(1^\kappa)$**:** The key generation algorithm takes the security parameter 1^κ as input, and works as follows:

1. Generate $(\mathsf{svk}, \mathsf{ssk}) \xleftarrow{\$} \mathsf{SigGen}(1^\kappa)$.
2. Generate $\mathsf{com} \xleftarrow{\$} \mathsf{Com}(\mathsf{ssk}; r_{\mathsf{com}})$.
3. Generate $(\mathsf{pp}, \mathsf{sp}) \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$.
4. Output a public key $\mathsf{pk} := (\mathsf{svk}, \mathsf{com}, \mathsf{pp})$ and a signing key $\mathsf{sk} := (\mathsf{ssk}, r_{\mathsf{com}}, \mathsf{sp})$.

$\mathcal{U}_1(\mathsf{pk}, m)$**:** The user's first message generation algorithm takes as input a public key $\mathsf{pk} = (\mathsf{svk}, \mathsf{com}, \mathsf{pp})$ and a message m, and works as follows:

1. Generate $(\mathsf{sfe}_1, \mathsf{sfest}) \xleftarrow{\$} \mathsf{Receiver}(1^\kappa, m)$.
2. Generate $\mathsf{ch} \xleftarrow{\$} \mathcal{V}_1(\mathsf{pp})$.
3. Output a first message $\mu := (\mathsf{sfe}_1, \mathsf{ch})$ and a state $\mathsf{st}_{\mathcal{U}} := \mathsf{sfest}$.

$\mathcal{S}_2(\mathsf{sk}, \mu)$**:** The signer's second message generation algorithm takes as input a signing key $\mathsf{sk} = (\mathsf{ssk}, r_{\mathsf{com}}, \mathsf{sp})$. and a first message $\mu = (\mathsf{sfe}_1, \mathsf{ch})$ and works as follows:

1. Generate $\mathsf{sfe}_2 \xleftarrow{\$} \mathsf{Sender}(1^\kappa, \mathsf{sfe}_1, \mathsf{Sign}(\mathsf{ssk}, \cdot); r_{\mathsf{sfe}})$.
2. Generate $\mathsf{resp} \xleftarrow{\$} \mathcal{P}_2(\mathsf{sp}, \mathsf{ch}, (\mathsf{com}, \mathsf{sfe}_1, \mathsf{sfe}_2), (\mathsf{ssk}, r_{\mathsf{com}}, r_{\mathsf{sfe}}))$.

3. Output a second message $\rho := (\mathsf{sfe}_2, \mathsf{resp})$.

$\mathcal{U}_{\mathsf{der}}(\mathsf{st}_{\mathcal{U}}, \rho)$**:** The user's signature derivation algorithm takes as input a state $\mathsf{st}_{\mathcal{U}} = \mathsf{sfest}$ and a second message $\rho = (\mathsf{sfe}_2, \mathsf{resp})$ as input, and works as follows:

1. Output $\bot$ if $\mathcal{V}_{\mathsf{out}}(\mathsf{pp}, \mathsf{ch}, (\mathsf{com}, \mathsf{sfe}_1, \mathsf{sfe}_2), \mathsf{resp}) = \bot$
2. Otherwise generate $\sigma \xleftarrow{\$} \mathsf{Derive}(\mathsf{sfest}, \mathsf{sfe}_2)$ and output a signature σ.

$\mathsf{BSVerify}(\mathsf{pk}, m, \sigma)$**:** The verification algorithm takes as input a public key $\mathsf{pk} = (\mathsf{svk}, \mathsf{com}, \mathsf{pp})$, a message m, and a signature σ as input, and outputs $\mathsf{SigVerify}(\mathsf{svk}, m, \sigma)$.

The correctness of the scheme immediately follows from the correctness of Π_{Sig}, Π_{ZK}, and Π_{SFE}.

5.2 Security

In this section, we give security proofs for the above scheme.

Unforgeability.

Theorem 5.1. *If Π_{Sig} satisfies unforgeability against QPT adversaries,* Com *satisfies computational hiding against QPT adversaries, Π_{SFE} satisfies quantum-extraction sender's security against QPT adversaries, and Π_{ZK} satisfies reusable quantum-simulation zero-knowledge against classical adversaries, then Π_{BS} satisfies unforgeability against classical adversaries.*

Proof. We consider the following sequence of games between a PPT adversary $\mathcal{A}$ and a challenger. We denote by E_i the event that Game i returns 1.

Game 1**:** This is the original unforgeability game. That is, this game proceeds as follows.

1. The challenger generates $(\mathsf{ssk}, \mathsf{svk}) \xleftarrow{\$} \mathsf{SigGen}(1^\kappa)$, $\mathsf{com} \xleftarrow{\$} \mathsf{Com}(\mathsf{ssk}; r_{\mathsf{com}})$, and $(\mathsf{pp}, \mathsf{sp}) \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$, and defines a public key $\mathsf{pk} := (\mathsf{svk}, \mathsf{com}, \mathsf{pp})$ and a signing key $\mathsf{sk} := (\mathsf{ssk}, r_{\mathsf{com}}, \mathsf{sp})$, and sends pk to $\mathcal{A}$.
2. $\mathcal{A}$ can make arbitrarily many signing queries. When it makes a signing query $\mu = (\mathsf{sfe}_1, \mathsf{ch})$, the challenger generates $\mathsf{sfe}_2 \xleftarrow{\$} \mathsf{Sender}(1^\kappa, \mathsf{sfe}_1, \mathsf{Sign}(\mathsf{ssk}, \cdot); r_{\mathsf{sfe}})$ and $\mathsf{resp} \xleftarrow{\$} \mathcal{P}_2(\mathsf{sp}, \mathsf{ch}, (\mathsf{com}, \mathsf{sfe}_1, \mathsf{sfe}_2), (\mathsf{ssk}, r_{\mathsf{com}}, r_{\mathsf{sfe}}))$, and returns $\rho := (\mathsf{sfe}_2, \mathsf{resp})$.
3. Finally, $\mathcal{A}$ returns $\{(m_i, \sigma_i)\}_{i \in [q+1]}$ where q is the number of signing queries made by $\mathcal{A}$.

The game returns 1 if and only if $\mathcal{A}$ wins, i.e., $\{m_i\}_{i \in [q+1]}$ is pairwise distinct and $\mathsf{SigVerify}(\mathsf{svk}, m_i, \sigma_i) = \top$ for all $i \in [q+1]$. Our goal is to prove $\Pr[\mathsf{E}_1] = \mathsf{negl}(\kappa)$.

Game 2**:** This game is identical to the previous one except that resp is generated as $\mathsf{resp} \xleftarrow{\$} \mathcal{S}(\mathsf{pp}, \mathsf{ch}, (\mathsf{com}, \mathsf{sfe}_1, \mathsf{sfe}_2), 1^{|w|})$ when responding to each signing query where $\mathcal{S}$ is the simulator of Π_{ZK} and $|w|$ denotes the bit-length of $(\mathsf{ssk}, r_{\mathsf{com}}, r_{\mathsf{sfe}})$.

By a straightforward reduction to reusable quantum-simulation zero-knowledge property of Π_{ZK}, we have $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_1]| = \mathsf{negl}(\kappa)$.

Game 3**:** This game is identical to the previous one except that sfe_2 is generated as $m \xleftarrow{\$} \mathsf{SFEExt}(\mathsf{sfe}_1)$ and $\mathsf{sfe}_2 \xleftarrow{\$} \mathsf{SFESim}(1^\kappa, \mathsf{sfe}_1, \mathsf{Sign}(\mathsf{ssk}, m))$ when responding to each signing query.

Noting that r_{sfe} is no longer used for generating resp due to the modification made in Game 2, a straightforward reduction to quantum-extraction sender's security of Π_{SFE} gives $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\kappa)$. We note that the reduction works even though these games involve QPT computations (for $\mathcal{S}$ and SFEExt) since we assume quantum-extraction sender's security against *quantum* adversaries.

Game 4: In this game, the challenger generates com as $\mathsf{com} \xleftarrow{\$} \mathsf{Com}(0^{|\mathsf{ssk}|})$.

Noting that r_{com} is no longer used for generating resp due to the modification made in Game 2, a straightforward reduction to computational hiding of Π_{SFE} gives $|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_3]| = \mathsf{negl}(\kappa)$. We note that the reduction works even though these games involve QPT computations (for $\mathcal{S}$ and SFEExt) since we assume computational hiding against *quantum* adversaries.

What is left is to prove $\Pr[\mathsf{E}_4] = \mathsf{negl}(\kappa)$. We show this by considering the following QPT adversary $\mathcal{B}$ against unforgeability of Π_{Sig}.

$\mathcal{B}^{\mathsf{Sign}(\mathsf{ssk},\cdot)}(\mathsf{svk})$: It generates $\mathsf{com} \xleftarrow{\$} \mathsf{Com}(0^{|\mathsf{ssk}|})$ and $(\mathsf{pp}, \mathsf{sp}) \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$ and gives a public key $\mathsf{pk} := (\mathsf{svk}, \mathsf{com}, \mathsf{pp})$ to $\mathcal{A}$. When $\mathcal{A}$ makes a signing query $\mu = (\mathsf{sfe}_1, \mathsf{ch})$, $\mathcal{B}$ computes $m \xleftarrow{\$} \mathsf{SFEExt}(\mathsf{sfe}_1)$ and queries m to its own signing oracle to obtain $\sigma = \mathsf{Sign}(\mathsf{ssk}, m)$. Then $\mathcal{B}$ generates $\mathsf{sfe}_2 \xleftarrow{\$} \mathsf{SFESim}(1^\kappa, \mathsf{sfe}_1, \sigma)$ and $\mathsf{resp} \xleftarrow{\$} \mathcal{S}(\mathsf{pp}, \mathsf{ch}, (\mathsf{com}, \mathsf{sfe}_1, \mathsf{sfe}_2), 1^{|w|})$, and returns $\rho := (\mathsf{sfe}_2, \mathsf{resp})$ to $\mathcal{A}$ as a response from the signing oracle. Let $\{(m_i, \sigma_i)\}_{i\in[q+1]}$ be $\mathcal{A}$'s final output. $\mathcal{B}$ finds $i^* \in [q+1]$ such that it has not queried m_{i^*} to its own signing oracle and $\mathsf{SigVerify}(\mathsf{svk}, m_{i^*}, \sigma_{i^*}) = \top$, and outputs (m_{i^*}, σ_{i^*}). If there does not exist such i^*, $\mathcal{B}$ aborts.

It is easy to see that $\mathcal{B}$ perfectly simulates the environment of Game 4 to $\mathcal{A}$, and when $\mathcal{A}$ wins, $\mathcal{B}$ also wins (i.e., it succeeds in outputting (m_{i^*}, σ_{i^*}) such that $\mathsf{SigVerify}(\mathsf{svk}, m_{i^*}, \sigma_{i^*}) = \top$ and $\mathcal{B}$ has not queried m_{i^*}). Therefore, by unforgeability of Π_{Sig}, we have $\Pr[\mathsf{E}_4] = \mathsf{negl}(\kappa)$. This completes the proof of Theorem 5.1. □

Blindness

Theorem 5.2. *If* Com *satisfies perfect binding,* Π_{SFE} *satisfies receiver's security against non-uniform PPT adversaries, and* Π_{ZK} *satisfies adaptive soundness with untrusted setup against non-uniform PPT adversaries, then* Π_{BS} *satisfies blindness against PPT adversaries.*

Proof. We consider the following sequence of games between a PPT adversary $\mathcal{A}$ against the blindness and a challenger. We denote by E_i the event that Game i returns 1.

Game 1: This is the original blindness game. That is, this game proceeds as follows:

1. $\mathcal{A}$ is given as input the security parameter 1^κ, and sends a public key $\mathsf{pk} = (\mathsf{svk}, \mathsf{com}, \mathsf{pp})$ and a pair (m_0, m_1) of messages to the challenger.
2. The challenger generates $(\mathsf{sfe}_{1,b}, \mathsf{sfest}_b) \xleftarrow{\$} \mathsf{Receiver}(1^\kappa, m_b)$ and $\mathsf{ch}_b \xleftarrow{\$} \mathcal{V}_1(\mathsf{pp})$ and defines $\mu_b := (\mathsf{sfe}_{1,b}, \mathsf{ch}_b)$ and $\mathsf{st}_{\mathcal{U},b} := \mathsf{sfest}_b$ for each $b \in \{0,1\}$, picks $\mathsf{coin} \xleftarrow{\$} \{0,1\}$, and sends $(\mu_{\mathsf{coin}}, \mu_{1-\mathsf{coin}})$ to $\mathcal{A}$.
3. $\mathcal{A}$ sends $(\rho_{\mathsf{coin}} = (\mathsf{sfe}_{2,\mathsf{coin}}, \mathsf{resp}_{\mathsf{coin}}), \rho_{1-\mathsf{coin}} = (\mathsf{sfe}_{2,1-\mathsf{coin}}, \mathsf{resp}_{1-\mathsf{coin}}))$ to the challenger.
4. The challenger gives $(\bot, \bot)$ to $\mathcal{A}$ if $\mathcal{V}_{\mathsf{out}}(\mathsf{pp}, \mathsf{ch}_b, (\mathsf{com}, \mathsf{sfe}_{1,b}, \mathsf{sfe}_{2,b}), \mathsf{resp}) = \bot$ for either of $b \in \{0,1\}$. Otherwise it generates $\sigma_b \xleftarrow{\$} \mathsf{Derive}(\mathsf{sfest}_b, \mathsf{sfe}_{2,b})$ for each $b \in \{0,1\}$ and gives (σ_0, σ_1) to $\mathcal{A}$.
5. $\mathcal{A}$ outputs its guess coin'.

This game returns 1 if and only if $\mathsf{coin} = \mathsf{coin}'$. Our goal is to prove $\left|\Pr[\mathsf{E}_1] - \frac{1}{2}\right| = \mathsf{negl}(\kappa)$.

Game 2: This game is identical to the previous game except that we insert Step 1.5 between Step 1 and 2 and Step 4 is replaced with Step 4′ described below: (Differences of Step 4' from Step 4 are marked by red underlines.)

1.5.: The challenger finds $(\mathsf{ssk}, r_{\mathsf{com}})$ such that $\mathsf{com} = \mathsf{Com}(\mathsf{ssk}; r_{\mathsf{com}})$ by a brute-force search. If such $(\mathsf{ssk}, r_{\mathsf{com}})$ does not exist, it sets $(\mathsf{ssk}, r_{\mathsf{com}}) := (\bot, \bot)$.

4′.: The challenger gives $(\bot, \bot)$ to $\mathcal{A}$ if $\mathcal{V}_{\mathsf{out}}(\mathsf{pp}, \mathsf{ch}_b, (\mathsf{com}, \mathsf{sfe}_{1,b}, \mathsf{sfe}_{2,b}), \mathsf{resp}) = \bot$ for either of $b \in \{0,1\}$ or $(\mathsf{ssk}, r_{\mathsf{com}}) = (\bot, \bot)$. Otherwise it generates $\sigma_b := \mathsf{Sign}(\mathsf{ssk}, m_b)$ for each $b \in \{0,1\}$ and gives (σ_0, σ_1) to $\mathcal{A}$.

In Lemma 5.3, we prove $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_1]| = \mathsf{negl}(\kappa)$.

Game 3**:** This game is identical to the previous game except that $\mathsf{sfe}_{1,b}$ is generated as $(\mathsf{sfe}_{1,b}, \mathsf{sfest}_b) \xleftarrow{\$} \mathsf{Receiver}(1^\kappa, m_0)$ for both $b \in \{0,1\}$.

In Lemma 5.4, we prove $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\kappa)$.

Game 4**:** This game is identical to the previous game except that the challenger gives (μ_0, μ_1) to $\mathcal{A}$ instead of $(\mu_{\mathsf{coin}}, \mu_{1-\mathsf{coin}})$ in Step 2.

Since the distributions of μ_0 and μ_1 are identical, we have $\Pr[\mathsf{E}_4] = \Pr[\mathsf{E}_3]$. Moreover, since no information on coin is given to $\mathcal{A}$ in this game, we have $\Pr[\mathsf{E}_4] = \frac{1}{2}$.

What is left is to prove the following lemmata.

Lemma 5.3. *If* Com *satisfies perfect binding and* Π_{ZK} *satisfies adaptive soundness with untrusted setup against non-uniform PPT adversaries, then we have* $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_1]| = \mathsf{negl}(\kappa)$.

Proof. For each $b \in \{0,1\}$, we define Bad_b as an event that we have $\mathcal{V}_{\mathsf{out}}(\mathsf{pp}, \mathsf{ch}_b, (\mathsf{com}, \mathsf{sfe}_{1,b}, \mathsf{sfe}_{2,b}), \mathsf{resp}_b) = \top$ and

1. there does not exist $(\mathsf{ssk}, r_{\mathsf{com}})$ such that $\mathsf{com} = \mathsf{Com}(\mathsf{ssk}; r_{\mathsf{com}})$, or
2. there exists $(\mathsf{ssk}, r_{\mathsf{com}})$ such that $\mathsf{com} = \mathsf{Com}(\mathsf{ssk}; r_{\mathsf{com}})$ and $\mathsf{Derive}(\mathsf{sfest}_b, \mathsf{sfe}_{2,b}) \neq \mathsf{Sign}(\mathsf{ssk}, m_b)$.

Game 2 and Game 1 may be different only if Bad_0 or Bad_1 occurs. Therefore, it suffices to prove $\Pr[\mathsf{Bad}_b] = \mathsf{negl}(\kappa)$ for each $b \in \{0,1\}$. First, we prove $\Pr[\mathsf{Bad}_0] = \mathsf{negl}(\kappa)$ by considering a non-uniform PPT cheating prover $\mathcal{P}^* = (\mathcal{P}^*_{\mathsf{Setup}}, \mathcal{P}^*_2)$ against adaptive soundness with adaptive setup of Π_{ZK} as described below:

$\mathcal{P}^*_{\mathsf{Setup}}(1^\kappa)$**:** It runs the first stage of $\mathcal{A}(1^\kappa)$ to obtain $\mathsf{pk} = (\mathsf{svk}, \mathsf{com}, \mathsf{pp})$ and (m_0, m_1). It finds $(\mathsf{ssk}, r_{\mathsf{com}})$ such that $\mathsf{com} = \mathsf{Com}(\mathsf{ssk}; r_{\mathsf{com}})$ by a brute-force search. If such $(\mathsf{ssk}, r_{\mathsf{com}})$ does not exist, it sets $(\mathsf{ssk}, r_{\mathsf{com}}) := (\bot, \bot)$. It outputs pp and $\mathsf{st}_{\mathcal{P}^*} := (\mathsf{pk}, m_0, m_1, \mathsf{ssk}, r_{\mathsf{com}}, \mathsf{st}_{\mathcal{A}})$ where $\mathsf{st}_{\mathcal{A}}$ denotes the snapshot of $\mathcal{A}$ at this point.

$\mathcal{P}^*_2(\mathsf{st}_{\mathcal{P}^*}, \mathsf{ch})$**:** It parses $(\mathsf{pk}, m_0, m_1, \mathsf{ssk}, r_{\mathsf{com}}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathsf{st}_{\mathcal{P}^*}$, generates $(\mathsf{sfe}_{1,b}, \mathsf{sfest}_b) \xleftarrow{\$} \mathsf{Receiver}(1^\kappa, m_b)$ for each $b \in \{0,1\}$ and $\mathsf{ch}_1 \xleftarrow{\$} \mathcal{V}_1(\mathsf{pp})$, sets $\mathsf{ch}_0 := \mathsf{ch}$, and defines $\mu_b := (\mathsf{sfe}_{1,b}, \mathsf{ch}_b)$ for each $b \in \{0,1\}$, picks $\mathsf{coin} \xleftarrow{\$} \{0,1\}$, and sends $(\mu_{\mathsf{coin}}, \mu_{1-\mathsf{coin}})$ to $\mathcal{A}$ to run the second stage of $\mathcal{A}$ to obtain $(\rho_{\mathsf{coin}} = (\mathsf{sfe}_{2,\mathsf{coin}}, \mathsf{resp}_{\mathsf{coin}}), \rho_{1-\mathsf{coin}} = (\mathsf{sfe}_{2,1-\mathsf{coin}}, \mathsf{resp}_{1-\mathsf{coin}}))$. If Bad_0 occurs, then $\mathcal{P}^*_2$ outputs $(\mathsf{com}, \mathsf{sfe}_{1,0}, \mathsf{sfe}_{2,0})$ and resp_0.

We can see that $\mathcal{P}^*$ perfectly simulates Game 1 for $\mathcal{A}$ until the second stage of $\mathcal{A}$. Moreover, if Bad_0 occurs, we have $\mathcal{V}_{\mathsf{out}}(\mathsf{pp}, \mathsf{ch}_0, (\mathsf{com}, \mathsf{sfe}_{1,0}, \mathsf{sfe}_{2,0}), \mathsf{resp}_0) = \top$ and $(\mathsf{com}, \mathsf{sfe}_{1,0}, \mathsf{sfe}_{2,0}) \notin \mathcal{L}$ noting that Com is perfectly binding. Therefore, by the adaptive soundness with untrusted setup of Π_{ZK}, we have $\Pr[\mathsf{Bad}_0] = \mathsf{negl}(\kappa)$. We can prove $\Pr[\mathsf{Bad}_1] = \mathsf{negl}(\kappa)$ analogously. This completes a proof of Lemma 5.3. □

Lemma 5.4. *If* Π_{SFE} *satisfies receiver's security against non-uniform PPT adversaries, then we have* $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\kappa)$.

Proof. We prove this by considering a non-uniform PPT cheating adversary $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ against receiver's security of Π_{SFE} as described below:

$\mathcal{B}_0(1^\kappa)$**:** It runs the first stage of $\mathcal{A}(1^\kappa)$ to obtain $\mathsf{pk} = (\mathsf{svk}, \mathsf{com}, \mathsf{pp})$ and (m_0, m_1). It finds $(\mathsf{ssk}, r_{\mathsf{com}})$ such that $\mathsf{com} = \mathsf{Com}(\mathsf{ssk}; r_{\mathsf{com}})$ by a brute-force search. If such $(\mathsf{ssk}, r_{\mathsf{com}})$ does not exist, it sets $(\mathsf{ssk}, r_{\mathsf{com}}) := (\bot, \bot)$. It outputs (m_0, m_1) and $\mathsf{st}_{\mathcal{P}^*} := (\mathsf{pk}, m_0, m_1, \mathsf{ssk}, r_{\mathsf{com}}, \mathsf{st}_{\mathcal{A}})$ where $\mathsf{st}_{\mathcal{A}}$ denotes the snapshot of $\mathcal{A}$ at this point.

$\mathcal{B}_1(\mathsf{st}_{\mathcal{B}}, \mathsf{sfe}_1)$**:** It sets $\mathsf{sfe}_{1,1} := \mathsf{sfe}_1$, generates $(\mathsf{sfe}_{1,0}, \mathsf{sfest}_0) \xleftarrow{\$} \mathsf{Receiver}(1^\kappa, m_0)$ and $\mathsf{ch}_b \xleftarrow{\$} \mathcal{V}_1(\mathsf{pp})$ for $b \in \{0,1\}$, defines $\mu_b := (\mathsf{sfe}_{1,b}, \mathsf{ch}_b)$ for each $b \in \{0,1\}$, picks $\mathsf{coin} \xleftarrow{\$} \{0,1\}$, and sends $(\mu_{\mathsf{coin}}, \mu_{1-\mathsf{coin}})$ to $\mathcal{A}$ to run the second stage of $\mathcal{A}$ to obtain $(\rho_{\mathsf{coin}} = (\mathsf{sfe}_{2,\mathsf{coin}}, \mathsf{resp}_{\mathsf{coin}}), \rho_{1-\mathsf{coin}} = (\mathsf{sfe}_{2,1-\mathsf{coin}}, \mathsf{resp}_{1-\mathsf{coin}}))$. Then $\mathcal{B}_1$ gives $(\bot, \bot)$ to $\mathcal{A}$ if $\mathcal{V}_{\mathsf{out}}(\mathsf{pp}, \mathsf{ch}_b, (\mathsf{com}, \mathsf{sfe}_{1,b}, \mathsf{sfe}_{2,b}), \mathsf{resp}) = \bot$ for either of $b \in \{0,1\}$ or $(\mathsf{ssk}, r_{\mathsf{com}}) = (\bot, \bot)$. Otherwise it generates $\sigma_b := \mathsf{Sign}(\mathsf{ssk}, m_b)$ for each $b \in \{0,1\}$ and gives (σ_0, σ_1) to $\mathcal{A}$. Let coin' be $\mathcal{A}$'s final output. $\mathcal{B}_1$ outputs 1 if $\mathsf{coin} = \mathsf{coin}'$.

Clearly, $\mathcal{B}$ perfectly simulates Game 3 (resp. Game 2) if sfe_1 is generated as $(\mathsf{sfe}_1, \mathsf{sfest}) \xleftarrow{\$} \mathsf{Receiver}(1^\kappa, m_0)$ (resp. $(\mathsf{sfe}_1, \mathsf{sfest}) \xleftarrow{\$} \mathsf{Receiver}(1^\kappa, m_1)$). Therefore, by receiver's security of Π_{SFE}, we have $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\kappa)$. □

Combining the above, Theorem 5.2 is proven. □

5.3 Instantiations

As mentioned in Section 2 and Section 4.4, we can instantiate Π_{Sig} and Com under the QPT hardness of LWE, Π_{SFE} under the existence of one-way function against QPT adversaries and the non-uniform PPT hardness of DDH, DLIN, QR, or DCR, and Π_{ZK} under the QPT hardness of LWE and non-uniform PPT hardness of either of QR, DCR, SXDH in pairing groups, or DLIN in pairing groups. Noting that the LWE assumption implies the existence of one-way functions and the SXDH assumption implies the DDH assumption, we can instantiate all the building blocks if we assume the QPT hardness of LWE and non-uniform PPT hardness of either of QR, DCR, SXDH in pairing groups, or DLIN in pairing groups.

Acknowledgement. We thank anonymous reviewers of Eurocrypt 2021 for their helpful comments. The first and third authors were supported by JST CREST Grant Number JPMJCR19F6. The third author is also supported by JSPS KAKENHI Grant Number 19H01109.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Heidelberg, May / June 2010. (Cited on page 12.)

[Abe01] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 136–151. Springer, Heidelberg, May 2001. (Cited on page 8.)

[AFG+16] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. *Journal of Cryptology*, 29(2):363–421, April 2016. (Cited on page 2.)

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of Mathematics*, 160:781–793, 2004. (Cited on page 15.)

[AO12] Masayuki Abe and Miyako Ohkubo. A framework for universally composable non-committing blind signatures. *Int. J. Appl. Cryptogr.*, 2(3):229–249, 2012. (Cited on page 2.)

[AR04] Dorit Aharonov and Oded Regev. Lattice problems in NP cap coNP. In *45th FOCS*, pages 362–371. IEEE Computer Society Press, October 2004. (Cited on page 8.)

[AR05] Dorit Aharonov and Oded Regev. Lattice problems in NP cap conp. *J. ACM*, 52(5):749–765, 2005. (Cited on page 8, 17, 40.)

[BBK+16] Nir Bitansky, Zvika Brakerski, Yael Tauman Kalai, Omer Paneth, and Vinod Vaikuntanathan. 3-message zero knowledge against human ignorance. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 57–83. Springer, Heidelberg, October / November 2016. (Cited on page 9.)

[BCC04] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004*, pages 132–145. ACM Press, October 2004. (Cited on page 1.)

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th ACM STOC*, pages 505–514. ACM Press, May / June 2014. (Cited on page 9.)

[BFPV11] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 403–422. Springer, Heidelberg, March 2011. (Cited on page 2.)

[BGI+17] Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 275–303. Springer, Heidelberg, December 2017. (Cited on page 9.)

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, Heidelberg, April 2009. (Cited on page 5, 11.)

[BKP19] Nir Bitansky, Dakshita Khurana, and Omer Paneth. Weak zero-knowledge beyond the black-box barrier. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1091–1102. ACM Press, June 2019. (Cited on page 8.)

[BL18] Nir Bitansky and Huijia Lin. One-message zero knowledge and non-malleable commitments. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 209–234. Springer, Heidelberg, November 2018. (Cited on page 9.)

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013. (Cited on page 39.)

[BNPS02] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The power of RSA inversion oracles and the security of Chaum's RSA-based blind signature scheme. In Paul F. Syverson, editor, *FC 2001*, volume 2339 of *LNCS*, pages 319–338. Springer, Heidelberg, February 2002. (Cited on page 2.)

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, January 2003. (Cited on page 2.)

[BP04] Boaz Barak and Rafael Pass. On the possibility of one-message weak zero-knowledge. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 121–132. Springer, Heidelberg, February 2004. (Cited on page 9.)

[BPV12] Olivier Blazy, David Pointcheval, and Damien Vergnaud. Compact round-optimal partially-blind signatures. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 95–112. Springer, Heidelberg, September 2012. (Cited on page 2.)

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982. (Cited on page 1.)

[Cha88] David Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 177–182. Springer, Heidelberg, May 1988. (Cited on page 1.)

[CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of Cryptology*, 25(4):601–639, October 2012. (Cited on page 12.)

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, May 2001. (Cited on page 1.)

[DFKS16] Nico Döttling, Nils Fleischhacker, Johannes Krupp, and Dominique Schröder. Two-message, oblivious evaluation of cryptographic functionalities. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 619–648. Springer, Heidelberg, August 2016. (Cited on page 8.)

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *41st FOCS*, pages 283–293. IEEE Computer Society Press, November 2000. (Cited on page 3.)

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007. (Cited on page 3, 11.)

[FHKS16] Georg Fuchsbauer, Christian Hanser, Chethan Kamath, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model from weaker assumptions. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 391–408. Springer, Heidelberg, August / September 2016. (Cited on page 2.)

[FHS15] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 233–253. Springer, Heidelberg, August 2015. (Cited on page 2, 8.)

[Fis06] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, Heidelberg, August 2006. (Cited on page 2.)

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st FOCS*, pages 308–317. IEEE Computer Society Press, October 1990. (Cited on page 4.)

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1):1–28, 1999. (Cited on page 4, 11.)

[FOO93] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *AUSCRYPT'92*, volume 718 of *LNCS*, pages 244–251. Springer, Heidelberg, December 1993. (Cited on page 1.)

[FS10] Marc Fischlin and Dominique Schröder. On the impossibility of three-move blind signature schemes. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 197–215. Springer, Heidelberg, May / June 2010. (Cited on page 2, 8.)

[GG14] Sanjam Garg and Divya Gupta. Efficient round optimal blind signatures. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 477–495. Springer, Heidelberg, May 2014. (Cited on page 2, 3.)

[Gha17] Essam Ghadafi. Efficient round-optimal blind signatures in the standard model. In Aggelos Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 455–473. Springer, Heidelberg, April 2017. (Cited on page 2.)

[GLN11] Oded Goldreich, Leonid A. Levin, and Noam Nisan. On constructing 1-1 one-way functions. In Oded Goldreich, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, volume 6650 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 2011. (Cited on page 15.)

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994. (Cited on page 3.)

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012. (Cited on page 11.)

[GRS+11] Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh. Round optimal blind signatures. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 630–648. Springer, Heidelberg, August 2011. (Cited on page 2, 3, 4, 8, 12, 14, 25.)

[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. (Cited on page 3.)

[HK12] Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, 25(1):158–193, January 2012. (Cited on page 13, 36, 37.)

[HK16] Lucjan Hanzlik and Kamil Kluczniak. A short paper on blind signatures from knowledge assumptions. In Jens Grossklags and Bart Preneel, editors, *FC 2016*, volume 9603 of *LNCS*, pages 535–543. Springer, Heidelberg, February 2016. (Cited on page 8.)

[HKKL07] Carmit Hazay, Jonathan Katz, Chiu-Yuen Koo, and Yehuda Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 323–341. Springer, Heidelberg, February 2007. (Cited on page 2, 8.)

[HKL19] Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 345–375. Springer, Heidelberg, May 2019. (Cited on page 8.)

[HLOV11] Brett Hemenway, Benoît Libert, Rafail Ostrovsky, and Damien Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 70–88. Springer, Heidelberg, December 2011. (Cited on page 11.)

[JKKR17] Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Ron Rothblum. Distinguisher-dependent simulation in two rounds and its applications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 158–189. Springer, Heidelberg, August 2017. (Cited on page 9.)

[KK19] Yael Tauman Kalai and Dakshita Khurana. Non-interactive non-malleability from quantum supremacy. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 552–582. Springer, Heidelberg, August 2019. (Cited on page 2, 3, 9.)

[KRR14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In David B. Shmoys, editor, *46th ACM STOC*, pages 485–494. ACM Press, May / June 2014. (Cited on page 9.)

[Lin08] Yehuda Lindell. Lower bounds and impossibility results for concurrent self composition. *Journal of Cryptology*, 21(2):200–249, April 2008. (Cited on page 2, 8.)

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009. (Cited on page 37.)

[LS19] Alex Lombardi and Luke Schaeffer. A note on key agreement and non-interactive commitments. Cryptology ePrint Archive, Report 2019/279, 2019. https://eprint.iacr.org/2019/279. (Cited on page 10.)

[LVW20] Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. Statistical ZAPR arguments from bilinear maps. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 620–641. Springer, Heidelberg, May 2020. (Cited on page 13, 36, 37.)

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Heidelberg, April 2012. (Cited on page 39.)

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004. (Cited on page 39.)

[MSF10] Sarah Meiklejohn, Hovav Shacham, and David Mandell Freeman. Limitations on transformations from composite-order to prime-order groups: The case of round-optimal blind signatures. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 519–538. Springer, Heidelberg, December 2010. (Cited on page 2.)

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *12th SODA*, pages 448–457. ACM-SIAM, January 2001. (Cited on page 13, 36, 37.)

[Oka06] Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 80–99. Springer, Heidelberg, March 2006. (Cited on page 8.)

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 160–176. Springer, Heidelberg, May 2003. (Cited on page 3, 9.)

[Pas11] Rafael Pass. Limits of provable security from standard assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 109–118. ACM Press, June 2011. (Cited on page 2, 8.)

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342. ACM Press, May / June 2009. (Cited on page 39.)

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2008. (Cited on page 5, 11.)

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. (Cited on page 8.)

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009. (Cited on page 8, 10, 17, 39, 40.)

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990. (Cited on page 12.)

[SC12] Jae Hong Seo and Jung Hee Cheon. Beyond the limitation of prime-order bilinear groups, and round optimal blind signatures. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 133–150. Springer, Heidelberg, March 2012. (Cited on page 2.)

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994. (Cited on page 15, 37.)

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. (Cited on page 37, 38.)

A Construction of Secure Function Evaluation

In this section, we give a construction of SFE that satisfies the security properties defined in Section 2.6 based on an oblivious transfer (OT) with certain properties and a one-way function against quantum adversaries. We note that the former assumption can be instantiated based on the non-uniform PPT hardness of DDH [NP01], DLIN [LVW20], quadratic residuosity (QR), or decisional composite residuosity (DCR) [HK12] problems and the latter assumption can be instantiated based on QPT hardness of the LWE problem. Therefore, our SFE can be instantiated based on the non-uniform classical hardness of either of DDH, DLIN, QR, or DCR problems and the quantum hardness of the LWE problem.

A.1 Building Blocks

Here, we prepare building blocks for our construction of SFE.

Oblivious Transfer. The first building block is oblivious transfer (OT). An OT protocol with a message space $\mathcal{M}$ consists of PPT algorithms $(\mathsf{OT.Receiver}, \mathsf{OT.Sender}, \mathsf{OT.Derive})$.

$\mathsf{OT.Receiver}(1^\kappa, \beta)$**:** This is an algorithm supposed to be run by a receiver that takes the security parameter 1^κ and $\beta \in \{0,1\}$ as input and outputs a first message ot_1 and a receiver's state otst.

$\mathsf{OT.Sender}(1^\kappa, \mathsf{ot}_1, m_0, m_1)$**:** This is an algorithm supposed to be run by a sender that takes the security parameter 1^κ, a first message ot_1 sent from a receiver and a pair of two messages $(m_0, m_1) \in \mathcal{M}^2$ as input and outputs a second message ot_2.

$\mathsf{OT.Derive}(\mathsf{otst}, \mathsf{ot}_2)$**:** This is an algorithm supposed to be run by a receiver that takes a receiver's state otst and a second message ot_2 as input and outputs a message $m' \in \mathcal{M}$.

As correctness, we require the following:

Correctness. For any $\kappa \in \mathbb{N}$, $\beta \in \{0,1\}$, and $(m_0, m_1) \in \mathcal{M}^2$, we have

$$\Pr[\mathsf{OT.Derive}(\mathsf{otst}, \mathsf{ot}_2) = m_\beta : (\mathsf{ot}_1, \mathsf{otst}) \xleftarrow{\$} \mathsf{OT.Receiver}(1^\kappa, \beta), \mathsf{ot}_2 \xleftarrow{\$} \mathsf{OT.Sender}(1^\kappa, \mathsf{ot}_1, (m_0, m_1))] = 1.$$

Security requirements are essentially the same as those for statistically-sender-private OT [NP01] except that we require that the extraction algorithm is QPT instead of unbounded time. Specifically, we require the following two security notions.

Receiver's Security against Non-Uniform PPT Adversary. For any non-uniform PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have

$$\left| \Pr\left[\mathcal{A}_1(\mathsf{st}, \mathsf{ot}_1) = 1 : \begin{array}{l} \mathsf{st} \xleftarrow{\$} \mathcal{A}_0(1^\kappa) \\ (\mathsf{ot}_1, \mathsf{otst}) \xleftarrow{\$} \mathsf{OT.Receiver}(1^\kappa, 0) \end{array} \right] \right.$$
$$\left. - \Pr\left[\mathcal{A}_1(\mathsf{st}, \mathsf{ot}_1) = 1 : \begin{array}{l} \mathsf{st} \xleftarrow{\$} \mathcal{A}_0(1^\kappa) \\ (\mathsf{ot}_1, \mathsf{otst}) \xleftarrow{\$} \mathsf{OT.Receiver}(1^\kappa, 1) \end{array} \right] \right| \leq \mathsf{negl}(\kappa).$$

Quantum-Extraction Statistical Sender's Security. There exists a QPT algorithm OTExt and a PPT algorithm OTSim that satisfy the following: For any unbounded-time adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have

$$|\Pr[\mathcal{A}_1(\mathsf{st}_\mathcal{A}, \mathsf{ot}_2) = 1 : (\mathsf{ot}_1, (m_0, m_1), \mathsf{st}_\mathcal{A}) \xleftarrow{\$} \mathcal{A}_0(1^\kappa), \mathsf{ot}_2 \xleftarrow{\$} \mathsf{OT.Sender}(1^\kappa, \mathsf{ot}_1, (m_0, m_1))]$$
$$- \Pr[\mathcal{A}_1(\mathsf{st}_\mathcal{A}, \mathsf{ot}_2) = 1 : (\mathsf{ot}_1, (m_0, m_1), \mathsf{st}_\mathcal{A}) \xleftarrow{\$} \mathcal{A}_0(1^\kappa), \beta \xleftarrow{\$} \mathsf{OTExt}(\mathsf{ot}_1), \mathsf{ot}_2 \xleftarrow{\$} \mathsf{OTSim}(1^\kappa, \mathsf{ot}_1, m_\beta)]| \leq \mathsf{negl}(\kappa).$$

An OT protocol that satisfies the above security requirements can be constructed based on the non-uniform PPT hardness of DDH [NP01], DLIN [LVW20], QR, or DCR [HK12] problems. Namely, these works give OT protocols that satisfy the same security requirements as above except that they allow OTExt to run unbounded-time instead of QPT. Given the fact that factoring and discrete logarithm can be solved in QPT [Sho94], we can see that these extractors actually run in QPT.

Yao's Garbling. Our second building block is Yao's garbling. Yao's garbling scheme consists of PPT algorithms $(\mathsf{Yao.Garble}, \mathsf{Yao.Eval})$.

$\mathsf{Yao.Garble}(1^\kappa, C)$**:** This algorithm takes the security parameter 1^κ and a description of a classical circuit C with n-bit input as input and outputs a garbled circuit $\widetilde{C}$ and labels $\{\mathsf{label}_{i,b}\}_{i\in[n],b\in\{0,1\}}$.

$\mathsf{Yao.Eval}(\widetilde{C}, \{\widetilde{\mathsf{label}}_i\}_{i\in[n]})$**:** This is a deterministic algorithm that takes a garbled circuit $\widetilde{C}$ and labels $\{\widetilde{\mathsf{label}}_i\}_{i\in[n]}$ as input and outputs an evaluation result y.

Correctness. For any $\kappa \in \mathbb{N}$, a classical circuit C with n-bit input, and $x \in \{0,1\}^n$ we have

$$\Pr[\mathsf{Yao.Eval}(\widetilde{C}, \{\mathsf{label}_{i,x_i}\}_{i\in[n]}) = C(x) : (\widetilde{C}, \{\mathsf{label}_{i,b}\}_{i\in[n],b\in\{0,1\}}) \xleftarrow{\$} \mathsf{Yao.Garble}(1^\kappa, C)] = 1.$$

Quantum Security. There exists a PPT simulator YaoSim such that for any classical circuit C with n-bit input, $x \in \{0,1\}^n$ and a QPT distinguisher $\mathcal{D}$, we have

$$\begin{aligned}&\Pr[\mathcal{D}(\widetilde{C}, \{\mathsf{label}_{i,x_i}\}_{i\in[n]}) = 1 : (\widetilde{C}, \{\mathsf{label}_{i,b}\}_{i\in[n],b\in\{0,1\}}) \xleftarrow{\$} \mathsf{Yao.Garble}(1^\kappa, C)]\\ &- \Pr[\mathcal{D}(\widetilde{C}, \{\widetilde{\mathsf{label}}_i\}_{i\in[n]}) = 1 : (\widetilde{C}, \{\widetilde{\mathsf{label}}_i\}_{i\in[n]}) \xleftarrow{\$} \mathsf{YaoSim}(1^\kappa, C(x))] = \mathsf{negl}(\kappa)\end{aligned}$$

It is well-known that Yao's garbling can be constructed based on the existence of one-way function [Yao86, LP09]. The security proof also works in the quantum setting, and thus we can construct secure Yao's garbling against quantum adversaries based on any one-way function against quantum adversaries. Especially, the LWE assumption against QPT adversaries suffices.

A.2 Construction

Here, we give a construction of SFE. The construction is Yao's two-party computation protocol [Yao86] instantiated based on OT and garbling that satisfies the security defined in Appendix A.1. Let $(\mathsf{OT.Receiver}, \mathsf{OT.Sender}, \mathsf{OT.Derive})$ be an OT protocol and $(\mathsf{Yao.Garble}, \mathsf{Yao.Eval})$ be Yao's garbling that satisfies the security defined in Appendix A.1 and we assume that the message space for the OT matches the label space of Yao's garbling. Then we construct SFE as follows:

$\mathsf{SFE.Receiver}(1^\kappa, x)$**:** Let n be the length of x. Then for each $i \in [n]$, it generates $(\mathsf{ot}_{i,1}, \mathsf{otst}_i) \xleftarrow{\$} \mathsf{OT.Receiver}(1^\kappa, x_i)$ where x_i denotes the i-th bit of x. Then it outputs $\mathsf{sfe}_1 := \{\mathsf{ot}_{i,1}\}_{i\in[n]}$ and $\mathsf{sfest} := \{\mathsf{otst}_i\}_{i\in[n]}$.

$\mathsf{SFE.Sender}(1^\kappa, \mathsf{sfe}_1, C)$**:** It parses $\{\mathsf{ot}_{i,1}\}_{i\in[n]} \leftarrow \mathsf{sfe}_1$, generates $(\widetilde{C}, \{\mathsf{label}_{i,b}\}_{i\in[n],b\in\{0,1\}}) \xleftarrow{\$} \mathsf{Yao.Garble}(1^\kappa, C)$ and $\mathsf{ot}_{i,2} \xleftarrow{\$} \mathsf{OT.Sender}(1^\kappa, \mathsf{ot}_{i,1}, \mathsf{label}_{i,0}, \mathsf{label}_{i,1})$ for all $i \in [n]$ and outputs $\mathsf{sfe}_2 := (\widetilde{C}, \{\mathsf{ot}_{i,2}\}_{i\in[n]})$

$\mathsf{SFE.Derive}(\mathsf{sfest}, \mathsf{sfe}_2)$**:** It parses $\{\mathsf{otst}_i\}_{i\in[n]} \leftarrow \mathsf{sfest}$, $(\widetilde{C}, \{\mathsf{ot}_{i,2}\}_{i\in[n]}) \leftarrow \mathsf{sfe}_2$, computes $\widetilde{\mathsf{label}}_i \xleftarrow{\$} \mathsf{OT.Derive}(\mathsf{otst}_i, \mathsf{ot}_{i,2})$ for all $i \in [n]$, and finally outputs $y := \mathsf{Yao.Eval}(\widetilde{C}, \{\widetilde{\mathsf{label}}_i\}_{i\in[n]})$

Correctness immediately follows from the correctness of underlying primitives. In the following, we give a proof sketch for security.

Receiver's Security against Non-Uniform PPT Adversary. Since the first message of SFE just consists of an n-tuple of first messages of the OT protocol, we can reduce the receiver's security of SFE to that of OT by a standard hybrid argument.

Quantum Extraction Sender's Security against QPT Adversary. For any QPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we consider the following sequence of games between $\mathcal{A}$ and a challenger where we denote by E_j the event that Game j returns 1:

Game 1**:** This game is the "real game" where $\mathcal{A}$'s inputs are simulated as in the first term of the inequality in the definition of quantum extraction sender's security in Section 2.6. That is, it is described as follows:

1. Run $(\mathsf{sfe}_1 = \{\mathsf{ot}_{i,1}\}_{i\in[n]}, C, \mathsf{st}_\mathcal{A}) \xleftarrow{\$} \mathcal{A}_0(1^\kappa)$.
2. Generate $(\widetilde{C}, \{\mathsf{label}_{i,b}\}_{i\in[n],b\in\{0,1\}}) \xleftarrow{\$} \mathsf{Yao.Garble}(1^\kappa, C)$ for all $i \in [n]$.
3. Generate $\mathsf{ot}_{i,2} \xleftarrow{\$} \mathsf{OT.Sender}(1^\kappa, \mathsf{ot}_{i,1}, \mathsf{label}_{i,0}, \mathsf{label}_{i,1})$ for all $i \in [n]$.
4. Set $\mathsf{sfe}_2 := (\widetilde{C}, \{\mathsf{ot}_{i,2}\}_{i\in[n]})$.

5. Return what $\mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, \mathsf{sfe}_2)$ outputs.

Game 2: This game is identical to the previous game except that we replace the Step 3 with the following Step 3'

3'. Run $x_i \xleftarrow{\$} \mathsf{OTExt}(\mathsf{ot}_{i,1})$ and $\mathsf{ot}_{i,2} \xleftarrow{\$} \mathsf{OTSim}(1^\kappa, \mathsf{ot}_{i,1}, \mathsf{label}_{i,x_i})$ for all $i \in [n]$.

We have $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_1]| \leq \mathsf{negl}(\kappa)$ by the quantum-extraction statistical sender's security of the OT.

Game 3: This game is identical to the previous game except that the computation of "$x_i \xleftarrow{\$} \mathsf{OTExt}(\mathsf{ot}_{i,1})$ for all $i \in [n]$" in Step 3' is done before Step 2. Since this is a conceptual change, we have $\Pr[\mathsf{E}_3] = \Pr[\mathsf{E}_2]$.

Game 4: This game is identical to the previous game except that we replace the Step 2 and Step 3' with the following Step 2' and Step 3" respectively

2'. Generate $(\widetilde{C}, \{\widetilde{\mathsf{label}}_i\}_{i\in[n]}) \xleftarrow{\$} \mathsf{YaoSim}(1^\kappa, C(x))$ where $x := x_1||...||x_n$.

3". Generate $\mathsf{ot}_{i,2} \xleftarrow{\$} \mathsf{OTSim}(1^\kappa, \mathsf{ot}_{i,1}, \widetilde{\mathsf{label}}_i)$ for all $i \in [n]$.

We note that this modification is possible since x_i for each $i \in [n]$ is generated before Step 2 due to the modification made in Game_3. We have $|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_3]| \leq \mathsf{negl}(\kappa)$ by a straightforward reduction to the security of Yao's garbling against quantum adversaries.

By combining the above, we have $|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_1]| \leq \mathsf{negl}(\kappa)$.

We can see that Game 4 corresponds to the "simulated game" where $\mathcal{A}$'s inputs are simulated as in the second term of the inequality in the definition of quantum extraction sender's security in Section 2.6. w.r.t. the following SFEExt and SFESim:

$\mathsf{SFEExt}(\mathsf{sfe}_1)$: It parses $\{\mathsf{ot}_{i,1}\}_{i\in[n]} \leftarrow \mathsf{sfe}_1$, computes $x_i \xleftarrow{\$} \mathsf{OTExt}(\mathsf{ot}_{i,1})$ for all $i \in [n]$, and output $x := x_1||...||x_n$.

$\mathsf{SFESim}(1^\kappa, \mathsf{sfe}_1, C(x))$: It parses $\{\mathsf{ot}_{i,1}\}_{i\in[n]} \leftarrow \mathsf{sfe}_1$, computes $(\widetilde{C}, \{\widetilde{\mathsf{label}}_i\}_{i\in[n]}) \xleftarrow{\$} \mathsf{YaoSim}(1^\kappa, C(x))$ and $\mathsf{ot}_{i,2} \xleftarrow{\$} \mathsf{OTSim}(1^\kappa, \mathsf{ot}_{i,1}, \widetilde{\mathsf{label}}_i)$ for all $i \in [n]$, and outputs $\mathsf{sfe}_2 := (\widetilde{C}, \{\mathsf{ot}_{i,2}\}_{i\in[n]})$.

We can see that SFEExt runs in QPT since OTExt runs in QPT and that SFESim runs in PPT since OTSim runs in PPT.

This completes the proof of quantum extraction sender's security against QPT adversaries.

B Construction of Public Key Encryption with Invalid Key Certifiability

In this section, we prove Lemma 3.5 by providing a PKE with invalid key certifiability based on the LWE problem. Below, we first prepare some tools on lattices. Note that we view vectors in the row form throughout this section.

B.1 Background on Lattices

A full-rank m-dimensional lattice Λ in $\mathbb{Z}^m$ is a set of the form $\{\sum_{i\in[m]} x_i\mathbf{b}_i | x_i \in \mathbb{Z}\}$, where $\mathbf{B} \in \mathbb{Z}^{m\times m}$ is the set of linearly independent row vectors $\{\mathbf{b}_1, \cdots, \mathbf{b}_m\}$. We call $\mathbf{B}$ the basis of the lattice Λ. For any matrix $\mathbf{A} \in \mathbb{Z}^{n\times m}$ such that $n \leq m$, define the full-rank m-dimensional lattice $\Lambda_q(\mathbf{A}) = \{\mathbf{b} \in \mathbb{Z}^m \mid \exists \mathbf{s} \in \mathbb{Z}_q^n \text{ s.t. } \mathbf{b} = \mathbf{sA} \mod q\}$. Using standard linear algebra, it is known that given $\mathbf{A}$, one can efficiently compute the basis $\mathbf{B}(\mathbf{A}) \in \mathbb{Z}^{m\times m}$ of the lattice $\Lambda_q(\mathbf{A})$. Finally, given a lattice $\Lambda \subseteq \mathbb{Z}^m$ and vector $\mathbf{b} \in \mathbb{Z}^m$, let $\mathsf{dist}(\Lambda, \mathbf{b})$ denote $\min_{\mathbf{v}\in\Lambda}\|\mathbf{v} - \mathbf{b}\|_2$.

Gaussian Measures. Let D_σ denote the discrete Gaussian distribution over $\mathbb{Z}$ with parameter $\sigma > 0$. For any positive integer n, D_σ^n is the discrete Gaussian distribution over $\mathbb{Z}^n$, where each coordinate is distributed independently according to D_σ. The following tail bound regarding D_σ^n holds.

Lemma B.1 ([MR04, Lyu12]). *We have* $\Pr[\mathbf{e} \xleftarrow{\$} D_\sigma^n : \|\mathbf{e}\|_2 > 2\sqrt{n}\sigma] < 2^{-n}$.

Hard Problems. We prepare two hard problems.

Definition B.2 (Learning with Errors). *For integers $n = n(\kappa), m = m(n)$, a prime $q = q(n) > 2$, an error distribution $\chi = \chi(n)$ over $\mathbb{Z}$, and a QPT $\mathcal{A}$, the advantage for the* learning with errors problem $\mathsf{LWE}_{n,m,q,\chi}$ *of $\mathcal{A}$ is defined as follows:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{LWE}_{n,m,q,\chi}} = \left| \Pr\left[\mathcal{A}(\mathbf{A}, \mathbf{sA} + \mathbf{e}) = 1\right] - \Pr\left[\mathcal{A}(\mathbf{A}, \mathbf{b}) = 1\right] \right|$$

where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{b} \xleftarrow{\$} \mathbb{Z}_q^m$, $\mathbf{e} \xleftarrow{\$} \chi^m$. We say that the LWE *assumption holds if $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{LWE}_{n,m,q,\chi}}$ is negligible for all QPT algorithm $\mathcal{A}$.*

The (decisional) $\mathsf{LWE}_{n,m,q,D_{\alpha q}}$ for $\alpha q > 2\sqrt{n}$ has been shown by Regev [Reg09] via a quantum reduction to be as hard as approximating the worst-case SIVP and GapSVP problems to within $\tilde{O}(n/\alpha)$ factors in the ℓ_2-norm in the worst case. In the subsequent works, (partial) dequantumization of the reduction were achieved [Pei09, BLP+13].

We also define a hard problem over lattices.

Definition B.3 (Gap Closest Vector Problem). *For a given parameter $\gamma > 1$ and $r > 0$, the promise problem $\mathsf{GapCVP}_{r,\gamma}$ is defined as follows. An input to the problem consists of a basis $\mathbf{B} \in \mathbb{Z}^{m \times m}$ of a lattice $\Lambda \subseteq \mathbb{Z}^m$ and a vector $\mathbf{b} \in \mathbb{Z}^m$. It is a* YES *instance if $\mathsf{dist}(\Lambda, \mathbf{b}) \leq r$ and a* NO *instance if $\mathsf{dist}(\Lambda, \mathbf{b}) > \gamma \cdot r$.*

Theorem B.4 ([AR05, Theorem 1.1]). *There exists a universal constant $c > 0$ and efficiently and deterministically computable relations $\mathcal{R}_{\mathsf{YES}}$ and $\mathcal{R}_{\mathsf{NO}}$ with binary output $\{\top, \bot\}$ satisfying the following.*

1. *If $(\mathbf{B}, \mathbf{b}) \in \mathbb{Z}^{m \times m} \times \mathbb{Z}^m$ is a* YES *instance to the $\mathsf{GapCVP}_{r, c\cdot\sqrt{m}}$ problem, then there exists a witness $\mathsf{wit}_{\mathsf{YES}}$ such that $\mathcal{R}_{\mathsf{YES}}((\mathbf{B}, \mathbf{b}), \mathsf{wit}_{\mathsf{YES}}) = \top$, and no witness $\mathsf{wit}_{\mathsf{NO}}$ such that $\mathcal{R}_{\mathsf{NO}}((\mathbf{B}, \mathbf{b}), \mathsf{wit}_{\mathsf{NO}}) = \top$.*
2. *If $(\mathbf{B}, \mathbf{b}) \in \mathbb{Z}^{m \times m} \times \mathbb{Z}^m$ is a* NO *instance to the $\mathsf{GapCVP}_{r, c\cdot\sqrt{m}}$ problem, there exists a witness $\mathsf{wit}_{\mathsf{NO}}$ such that $\mathcal{R}_{\mathsf{NO}}((\mathbf{B}, \mathbf{b}), \mathsf{wit}_{\mathsf{NO}}) = \top$, and no witness $\mathsf{wit}_{\mathsf{YES}}$ such that $\mathcal{R}_{\mathsf{YES}}((\mathbf{B}, \mathbf{b}), \mathsf{wit}_{\mathsf{YES}}) = \top$.*

B.2 Construction

We provide a construction of a PKE with invalid key certifiability based on lattices. The construction is essentially Regev's PKE scheme [Reg09] modulo some minor changes in the parameters. Below, let n, m, q, α be set so that the $\mathsf{LWE}_{n,m,q,D_{\alpha q}}$ problem is hard, i.e., $\alpha q > 2\sqrt{n}$, $4cm\sqrt{m}\alpha < 1$ for perfect correctness, and $m = 2n \log q$, where c is the constant associated to the GapCVP problem (see Theorem B.4). Moreover, let the bit length ℓ of the witness $\mathsf{wit}_{\mathsf{invalid}}$ used by algorithm $\mathsf{IKC.InvalidVerf}$ be equal to the witness length of the NO instance to the $\mathsf{GapCVP}_{r, c\sqrt{m}}$ problem, where $r = 2\sqrt{m}\alpha q$.

Construction. The construction of a PKE with invalid key certifiability is provided below.

$\mathsf{IKC.KeyGen}(1^\kappa)$: Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, and $\mathbf{e} \xleftarrow{\$} D_{\alpha q}^m$.[21] We assume without loss of generality that $\|\mathbf{e}\|_2 \leq 2\sqrt{m}\alpha q$. Set $\mathbf{b} = \mathbf{sA} + \mathbf{e}$ and output $\mathsf{ek}_{\mathsf{ikc}} = (\mathbf{A}, \mathbf{b})$ and $\mathsf{dk}_{\mathsf{ikc}} = \mathbf{s}$.

$\mathsf{IKC.Enc}(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{m})$**:** On input a message $\mathsf{m} \in \{0,1\}$, sample $\mathbf{r} \xleftarrow{\$} \{0,1\}^m$ and set $\mathbf{c}_0 = \mathbf{A}\mathbf{r}^\top$ and $c_1 = \mathbf{b}\mathbf{r}^\top + \mathsf{m} \cdot \lfloor q/2 \rceil$. Finally, output the ciphertext $\mathsf{ct}_{\mathsf{ikc}} = (\mathbf{c}_0, c_1)$.

$\mathsf{IKC.Dec}(\mathsf{dk}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{ikc}})$**:** On input a ciphertext $\mathsf{ct}_{\mathsf{ikc}} = (\mathbf{c}_0, c_1)$, compute $w = c_1 - \mathbf{s}\mathbf{c}_0$. Output 0 if w is closer to 0 than to $\lfloor q/2 \rceil$ modulo q. Otherwise output 1.

$\mathsf{IKC.InvalidVerf}(1^\kappa, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{wit}_{\mathsf{invalid}})$**:** On input an encryption key $\mathsf{ek}_{\mathsf{ikc}} = (\mathbf{A}, \mathbf{b})$ and a witness $\mathsf{wit}_{\mathsf{invalid}} \in \{0,1\}^\ell$, construct the basis $\mathbf{B}(\mathbf{A}) \in \mathbb{Z}^{m \times m}$ for the lattice $\Lambda_q(\mathbf{A})$. Then run $w \leftarrow \mathcal{R}_{\mathsf{NO}}((\mathbf{B}(\mathbf{A}), \mathbf{b}), \mathsf{wit}_{\mathsf{invalid}})$ by viewing $\mathbf{b}$ as a vector over $\mathbb{Z}^m$ and output the result w. Here note that since $q\mathbb{Z}^m \subseteq \Lambda_q(\mathbf{A})$, the choice of $\mathbf{b}$ is arbitrary.

[21] In case the complementary event happens (which occurs with probability less than 2^{-n} due to Lemma B.1), we simply choose $\mathbf{b}$ to be noise-free, i.e., $\mathbf{b} = \mathbf{sA}$.

Correctness. Perfect correctness follows from our parameter choice. Since we show that we have perfect correctness with respect to a "worse" encryption key $\mathsf{ek}_{\mathsf{ikc}}$ below, we omit the proof. Concretely, we consider the case where $\mathbf{b}$ is slightly further away from $\Lambda_q(\mathbf{A})$ compared to an honestly generated $(\mathbf{A}, \mathbf{b})$.

CPA-Security against Quantum Adversary. The proof is identical to Regev's PKE scheme. Informally, we rely on the (quantum secure) LWE problem to modify the public key to be indistinguishable from random matrices, and then invoke the leftover hash lemma to argue that the ciphertext is distributed uniformly random and independent from the message m.

Invalid Key Certifiability. We check that Items 1 and 2 regarding algorithm $\mathsf{IKC.InvalidVerf}$ hold.

We first check Item 1. Observe that for honestly generated keys $(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{dk}_{\mathsf{ikc}}) \xleftarrow{\$} \mathsf{IKC.KeyGen}(1^\kappa)$, we have $\mathsf{dist}(\Lambda_q(\mathbf{A}), \mathbf{b}) \leq \|\mathbf{e}\|_2 \leq 2\sqrt{m}\alpha q = r$ with overwhelming probability. Therefore, since $(\mathbf{B}(\mathbf{A}), \mathbf{b})$ is a YES instance to the $\mathsf{GapCVP}_{r, c\sqrt{m}}$ problem, there does not exist $\mathsf{wit}_{\mathsf{invalid}} \in \{0,1\}^\ell$ such that $\mathcal{R}_{\mathsf{NO}}((\mathbf{B}(\mathbf{A}), \mathbf{b}), \mathsf{wit}_{\mathsf{invalid}}) = \top$ due to Theorem B.4, Item 1. This in particular implies Item 1 by definition of $\mathsf{IKC.InvalidVerf}$.

We proceed to check Item 2. In case there does not exist $\mathsf{wit}_{\mathsf{invalid}} \in \{0,1\}^\ell$ such that $\mathsf{IKC.InvalidVerf}(1^\kappa, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{wit}_{\mathsf{invalid}}) = \top$, $(\mathbf{B}(\mathbf{A}), \mathbf{b})$ cannot be a NO instance due to Theorem B.4, Item 2. Namely, we have $\mathsf{dist}(\Lambda_q(\mathbf{A}), \mathbf{b}) \leq c\sqrt{m} \cdot r$. Therefore, there exists some $\mathbf{s}^* \in \mathbb{Z}_q^n$ such that $\|\mathbf{b} - \mathbf{s}^*\mathbf{A}\|_2 \leq c\sqrt{m} \cdot r$. Let us set any such $\mathbf{s}^*$ as the decryption key $\mathsf{dk}_{\mathsf{ikc}}$. Then for any $\mathsf{m} \in \{0,1\}$ and $(\mathbf{c}_0, c_1) = (\mathbf{A}\mathbf{r}^\top, \mathbf{b}\mathbf{r}^\top + \mathsf{m} \cdot \lfloor q/2 \rceil) \in \mathsf{IKC.Enc}(\mathsf{ek}_{\mathsf{ikc}}, \mathsf{m})$, we have $c_1 - \mathbf{s}^*\mathbf{c}_0 = (\mathbf{b} - \mathbf{s}^*\mathbf{A})\mathbf{r}^\top + \mathsf{m} \cdot \lfloor q/2 \rceil$. Since $\mathbf{r} \in \{0,1\}^m$, we have $\left|(\mathbf{b} - \mathbf{s}^*\mathbf{A})\mathbf{r}^\top\right| \leq \|\mathbf{b} - \mathbf{s}^*\mathbf{A}\|_2 \cdot \|\mathbf{r}\|_2 \leq cmr$. Since $cmr < q/2$ from our parameter choice, the decryption algorithm successfully outputs m. This holds for any $r \in \{0,1\}^m$. Hence, this completes the proof of Item 2.

Contents