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

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

[1] 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

**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[3], 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.[4] 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[5] 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?*

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

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

[5] 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.

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.[6]

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.

*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

[6] 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.

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.[7] 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)

[7] 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.

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.[8] 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 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 $r_{\mathsf{zap}}$ of the ZAP system along with a random image $z = f(y)$ of a one-way permutation (OWP) $f : \{0,1\}^\ell \to \{0,1\}^\ell$.[9]
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 $0^\ell$ by a non-interactive commitment with perfect biding to obtain $\mathsf{com} = \mathsf{Com}(0^\ell; r_{\mathsf{com}})$ using randomness $r_{\mathsf{com}}$. It then proves that there is witness $(w', y', r'_{\mathsf{com}})$ such that

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

[8] 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.

[9] 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.

by the proving algorithm of the ZAP system to obtain a proof $\pi_{\mathsf{zap}}$ 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 $\pi_{\mathsf{zap}}$ 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 $(r_{\mathsf{zap}}, z)$, 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 $(\mathsf{com}^*, \pi^*)$ of the successful adversary should satisfy the trapdoor branch of the relation. Namely, $\mathsf{com}^*$ 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 $\mathsf{com}^*$ 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 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 $0^\ell$ using a public key $\mathsf{pk}_P$ chosen by itself, instead of computing a commitment of $0^\ell$. The advice string in our context is the secret key corresponding to $\mathsf{pk}_P$. 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 $\mathsf{pk}_P$ *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 $\mathsf{pk}_P$ in the pre-computation phase using its unbounded computational power, since $\mathsf{pk}_P$ 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,[10] but there are no known PKE schemes satisfying these properties simultaneously.

[10] 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

To resolve the issue, we further change the protocol. Our first attempt is to let the verifier choose a public key $\mathsf{pk}_V$ of PKE and have the prover encrypt $0^\ell$ under $\mathsf{pk}_V$ in addition to the long-term public key $\mathsf{pk}_P$. Furthermore, we have the prover prove that it has valid witness $w$ for $x$ or it encrypts $y$ under $\mathsf{pk}_P$ and $\mathsf{pk}_V$. With this change, the reduction algorithm can extract the message from the ciphertext corresponding to $\mathsf{pk}_V$ even if $\mathsf{pk}_P$ is maliciously generated since $\mathsf{pk}_V$ 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 $\mathsf{pk}_V$, 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 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 $\mathsf{pk}_V$ 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.

---

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.

1. The prover runs the key generation algorithm of the PKE to obtain a public key $\mathsf{pk}_P$ and chooses the first message $r'_{\mathsf{zap}}$ 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 $r_{\mathsf{zap}}$ 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 $\mathsf{pk}_V$. It then proves statement (1) with respect to $r'_{\mathsf{zap}}$ by using the randomness for the lossy key generation as a witness to obtain a proof $\pi'_{\mathsf{zap}}$. Finally, it sends $(r_{\mathsf{zap}}, \mathsf{pk}_V, \pi'_{\mathsf{zap}})$ to the prover.
3. Given the message, the prover verifies $\pi'_{\mathsf{zap}}$ for statement (1) and aborts the protocol if it is not valid. Otherwise, it encrypts the string $0^\ell$ under $\mathsf{pk}_P$ and $\mathsf{pk}_V$ 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 $\mathsf{LE}$ 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 $r_{\mathsf{zap}}$ to obtain a proof $\pi_{\mathsf{zap}}$. 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 $\pi_{\mathsf{zap}}$ 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 $0^\ell$ when it generates the ciphertext $\mathsf{ct}_P$. 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 $\mathsf{ct}_V$ with the encryption of $y$ instead of $0^\ell$. 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 $\mathsf{pk}_P$ is honestly chosen, $\mathsf{pk}_V$ should be a lossy key. Then, by the lossiness of the lossy encryption, we conclude that $\mathsf{ct}_V$ does not carry any information about the message, and the change does not alter the distribution of $\mathsf{ct}_V$. Finally, we change the game so that the prover uses the latter branch of statement (2) to generate $\pi_{\mathsf{zap}}$. 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 $\mathsf{pk}_P$. In this case, the reduction algorithm receives $\mathsf{pk}_P$ from the malicious prover and finds the corresponding secret key $\mathsf{sk}_P$ 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, $\mathsf{ct}_P$ should be an encryption of $y = f^{-1}(z)$ under the public key $\mathsf{pk}_P$ and thus the reduction algorithm can successfully extract $y$ from $\mathsf{ct}_P$ by using $\mathsf{sk}_P$.

We next consider the other case where $\mathsf{pk}_P$ 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 $\mathsf{pk}_P$. 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 $\mathsf{pk}_V$ 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 $\mathsf{pk}_V$ 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 $\mathsf{pk}_P$ 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 $\mathsf{ct}_V$, which is possible because $\mathsf{pk}_V$ 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 $\mathsf{pk}_P$ 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 $(x^*, \pi^*)$ 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 $\pi^*$, $x^*$ 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 $\mathsf{ct}_P$ and $\mathsf{ct}_V$ 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 $\mathsf{pk}_P$ 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.

## 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 $1^\kappa$ 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.

Definitions of standard cryptographic primitives, including non-interactive commitment, public key encryption, lossy encryption, ZAP, and digital signatures, can be found in the full version.

### 2.1 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 $1^\kappa$ and $x$ as input and outputs a first message $\mathsf{sfe}_1$ and a receiver's state $\mathsf{sfest}$.

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

$\mathsf{Derive}(\mathsf{sfest}, \mathsf{sfe}_2) \to y$**:** This is an algorithm supposed to be run by a receiver that takes a receiver's state $\mathsf{sfest}$ and a second message $\mathsf{sfe}_2$ 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 $\mathsf{SFEExt}$ and a PPT algorithm $\mathsf{SFESim}$ that satisfy the following: For any QPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have

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

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], QR, or DCR [HK12]). See the full version for details.

### 2.2 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 $1^\kappa$ and outputs a public key $\mathsf{pk}$ and a signing key $\mathsf{sk}$.

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

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

$\mathsf{BSVerify}(\mathsf{pk}, m, \sigma) \to \top$ **or** $\bot$**:** This is a deterministic verification algorithm that takes as input a public key $\mathsf{pk}$, a message $m \in \mathcal{M}$, and a signature $\sigma$, 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 $1^\kappa$, and sends a public key $\mathsf{pk}$ 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 $(\sigma_0, \sigma_1)$ to $\mathcal{A}$.

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

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 $1^\kappa$ 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 $1^\kappa$ and a problem $\mathsf{prob}$ 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 $\mathsf{prob}$ and returns a solution $\mathsf{sol}$.

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

We say that $\Pi_{\mathsf{HPG}}$ 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 $\mathsf{prob}$ and its preimage as $\mathsf{sol}$. The efficient recognizability of the range is needed since otherwise we cannot implement $\mathsf{VerProb}$ 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 3.1.** *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.*

This is an easy consequence of Shor's algorithm [Sho94] that solves factoring and discrete logarithm in QPT. A full proof can be found in the full version.

### 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 $1^\kappa$, 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 3.2.* 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 3.3.* 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 3.2.** *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 ($\mathsf{GapCVP}$) 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 the full version.

## 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[11] and 2-move with reusable setup by the prover,[12]
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 $1^\kappa$ and outputs a public parameter $\mathsf{pp}$ and a secret parameter $\mathsf{sp}$.

$\mathcal{V}_1(\mathsf{pp}) \to \mathsf{ch}$**:** This is the verifier's first message generation algorithm that takes as input a public parameter $\mathsf{pp}$ and outputs a first message $\mathsf{ch}$ 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 $\mathsf{sp}$, a challenge $\mathsf{ch}$, a statement $x$, and a witness $w$, and outputs a second message $\mathsf{resp}$ 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 $\mathsf{pp}$, a challenge $\mathsf{ch}$, a statement $x$, and a response $\mathsf{resp}$, and outputs $\top$ to indicate acceptance or $\bot$ to indicate rejection.

It should satisfy the following properties:

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

[12] 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 $\perp$

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

### 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 $\Pi_{\mathsf{zap}}$ and $\Pi'_{\mathsf{zap}}$ are $\{0,1\}^\ell$, which can be assumed without loss of generality by taking $\ell$ as an arbitrarily large polynomial in $\kappa$. 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 $1^\kappa$, 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 $\perp$ if $\mathsf{VerProb}(1^\kappa, \mathsf{prob}) = \perp$ or $\mathsf{ZAP.Verify}'(r'_{\mathsf{zap}}, (\mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}), \pi'_{\mathsf{zap}}) = \perp$.
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 $\Pi_{\mathsf{zap}}$ and $\Pi'_{\mathsf{zap}}$ and the validity of an honestly generated instance of $\Pi_{\mathsf{HPG}}$.

### 4.3 Security

Here, we only give a proof sketch. A full proof can be found in the full version.

*Adaptive Soundness with Untrusted Setup.* Consider an interaction between an honest verifier and a cheating prover $\mathcal{P}^*$ (that may maliciously generate $\mathsf{pp}$). When $\mathcal{P}^*$ succeeds in breaking soundness, we have $x \notin \mathcal{L}$, which implies $(x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}) \notin \widetilde{\mathcal{L}}_1$. On the other hand, by soundness of $\Pi_{\mathsf{zap}}$, if the verifier accepts, then we have $(x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}) \in \widetilde{\mathcal{L}} = \widetilde{\mathcal{L}}_1 \cup \widetilde{\mathcal{L}}_2$ with overwhelming probability. Therefore, if $\mathcal{P}^*$ wins with non-negligible probability, we have $(x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}) \in \widetilde{\mathcal{L}}_2$. We assume that this happens and construct a reduction algorithm that breaks non-uniform PPT hardness of $\Pi_{\mathsf{HPG}}$. We consider the following two cases:

1. When $\mathsf{ek}_{\mathsf{ikc}}$ is decryptable (i.e., there is a corresponding decryption key $\mathsf{dk}_{\mathsf{ikc}}$ to $\mathsf{ek}_{\mathsf{ikc}}$): In this case, we can construct a reduction algorithm that finds $\mathsf{dk}_{\mathsf{ikc}}$ by brute-force and then extracts $\mathsf{sol}$ by decrypting $\mathsf{ct}_{\mathsf{ikc}}$ to break $\Pi_{\mathsf{HPG}}$. We note that the brute-force search can be done *before* getting a problem instance $\mathsf{prob}$, and thus non-uniform PPT hardness suffices.
2. When $\mathsf{ek}_{\mathsf{ikc}}$ is undecryptable: In this case, we consider several hybrids. In the first hybrid, $\pi'_{\mathsf{zap}}$ is generated by using a witness $\mathsf{wit}_{\mathsf{invalid}}$ instead of $r_{\mathsf{le\text{-}gen}}$. We note that such a witness $\mathsf{wit}_{\mathsf{invalid}}$ of invalidness of $\mathsf{ek}_{\mathsf{ikc}}$ must exist when $\mathsf{ek}_{\mathsf{ikc}}$ is undecryptable by the second property of invalid key certifiability. By witness indistinguishability of $\pi'_{\mathsf{zap}}$, this only negligibly changes the cheating prover's winning probability.[13] In the next hybrid, $\mathsf{ek}_{\mathsf{le}}$ is generated in the injective mode instead of lossy mode. By key indistinguishability of $\Pi_{\mathsf{LE}}$, this only negligibly changes the cheating prover's winning probability. At this point, a reduction algorithm can generate $\mathsf{ek}_{\mathsf{le}}$ in the injective mode with its corresponding decryption key, and thus it can extract $\mathsf{sol}$ by decrypting $\mathsf{ct}_{\mathsf{le}}$ to break $\Pi_{\mathsf{HPG}}$. Similarly to the previous case, the non-uniform PPT hardness suffices even though the reduction algorithm runs a brute-force algorithm to find $\mathsf{wit}_{\mathsf{invalid}}$ since this can be done *before* getting a problem instance $\mathsf{prob}$.

[13] Strictly speaking, since the event that the cheating prover wins is not efficiently checkable, a more careful analysis is needed.

This contradicts non-uniform PPT hardness of $\Pi_{\Pi_{\mathsf{HPG}}}$. Therefore, the cheating prover's winning probability is negligible, and thus soundness holds.

*Reusable Quantum-Simulation Zero-Knowledge.* A QPT simulator $\mathcal{S}$ is described as follows:

$\mathcal{S}(\mathsf{pp}, \mathsf{ch}, x, 1^{|w|})$**:** $\mathcal{S}$ 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 $\mathsf{ct}_{\mathsf{ikc}} \xleftarrow{\$} \mathsf{Enc}(\mathsf{sol}; r_{\mathsf{ikc\text{-}enc}})$ and $\mathsf{ct}_{\mathsf{le}} \xleftarrow{\$} \mathsf{LE.Enc}(\mathsf{ek}_{\mathsf{le}}, \mathsf{sol}; r_{\mathsf{le\text{-}enc}})$.
3. Generate $\mathsf{ct}_{\mathsf{pke}} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{ek}_{\mathsf{pke}}, 0^{|w|})$.
4. Generate $\pi_{\mathsf{zap}} \xleftarrow{\$} \mathsf{ZAP.Prove}(r_{\mathsf{zap}}, (x, \mathsf{ek}_{\mathsf{pke}}, \mathsf{ek}_{\mathsf{ikc}}, \mathsf{ek}_{\mathsf{le}}, \mathsf{prob}, \mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}), (\mathsf{sol}, r_{\mathsf{ikc\text{-}enc}}, r_{\mathsf{le\text{-}enc}}))$.
5. Return $\mathsf{resp} = (\mathsf{ct}_{\mathsf{pke}}, \mathsf{ct}_{\mathsf{ikc}}, \mathsf{ct}_{\mathsf{le}}, \pi_{\mathsf{zap}})$.

A response simulated by $\mathcal{S}$ is different from the real one in the following ways:

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

Roughly, the first difference is indistinguishable by the CPA security of $\Pi_{\mathsf{IKC}}$ 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 $\mathsf{sol}$ 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 $\Pi'_{\mathsf{ZAP}}$ 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 $\Pi_{\mathsf{ZAP}}$. The fourth difference is indistinguishable by the CPA security of $\Pi_{\mathsf{PKE}}$ against QPT adversaries after finishing the modification 3. 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.[14] 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 modifications 1, 2, and 3 for each query one-by-one similarly to [GRS+11]. In this way, we can ensure that all quantum computations can be done in 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.

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

## 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 $\mathsf{Sign}$ 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.
- $\mathsf{Com}$ 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 $1^\kappa$ 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 $\sigma$.

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

The correctness of the scheme immediately follows from the correctness of $\Pi_{\mathsf{Sig}}$, $\Pi_{\mathsf{ZK}}$, and $\Pi_{\mathsf{SFE}}$.

### 5.2 Security

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

*Unforgeability.*

**Theorem 5.1.** *If $\Pi_{\mathsf{Sig}}$ satisfies unforgeability against QPT adversaries,* Com *satisfies computational hiding against QPT adversaries, $\Pi_{\mathsf{SFE}}$ satisfies quantum-extraction sender's security against QPT adversaries, and $\Pi_{\mathsf{ZK}}$ satisfies reusable quantum-simulation zero-knowledge against classical adversaries, then $\Pi_{\mathsf{BS}}$ 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 $\mathsf{E}_i$ 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 $\mathsf{pk}$ 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 $\mathsf{resp}$ 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 $\Pi_{\mathsf{ZK}}$ 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 $\Pi_{\mathsf{ZK}}$, 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 $\mathsf{sfe}_2$ 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 $r_{\mathsf{sfe}}$ is no longer used for generating $\mathsf{resp}$ due to the modification made in Game 2, a straightforward reduction to quantum-extraction sender's security of $\Pi_{\mathsf{SFE}}$ 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 $\mathsf{SFEExt}$) since we assume quantum-extraction sender's security against *quantum* adversaries.

Game 4: In this game, the challenger generates $\mathsf{com}$ as $\mathsf{com} \xleftarrow{\$} \mathsf{Com}(0^{|\mathsf{ssk}|})$.
Noting that $r_{\mathsf{com}}$ is no longer used for generating $\mathsf{resp}$ due to the modification made in Game 2, a straightforward reduction to computational hiding of $\Pi_{\mathsf{SFE}}$ 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 $\mathsf{SFEExt}$) 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 $\Pi_{\mathsf{Sig}}$.

$\mathcal{B}^{\mathsf{Sign}(\mathsf{ssk},\cdot)}(\mathsf{svk})$: 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 $(m_{i^*}, \sigma_{i^*})$. 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 $(m_{i^*}, \sigma_{i^*})$ such that $\mathsf{SigVerify}(\mathsf{svk}, m_{i^*}, \sigma_{i^*}) = \top$ and $\mathcal{B}$ has not queried $m_{i^*}$). Therefore, by unforgeability of $\Pi_{\mathsf{Sig}}$, we have $\Pr[\mathsf{E}_4] = \mathsf{negl}(\kappa)$. This completes the proof of Theorem 5.1. □

*Blindness*

**Theorem 5.2.** *If* $\mathsf{Com}$ *satisfies perfect binding,* $\Pi_{\mathsf{SFE}}$ *satisfies receiver's security against non-uniform PPT adversaries, and* $\Pi_{\mathsf{ZK}}$ *satisfies adaptive soundness with untrusted setup against non-uniform PPT adversaries, then* $\Pi_{\mathsf{BS}}$ *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 $\mathsf{E}_i$ 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 $1^\kappa$, 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 $(\sigma_0, \sigma_1)$ to $\mathcal{A}$.
5. $\mathcal{A}$ outputs its guess $\mathsf{coin}'$.

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 $(\sigma_0, \sigma_1)$ to $\mathcal{A}$.

In Lemma 5.1, 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 5.2, 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 $(\mu_0, \mu_1)$ to $\mathcal{A}$ instead of $(\mu_{\mathsf{coin}}, \mu_{1-\mathsf{coin}})$ in Step 2.

Since the distributions of $\mu_0$ and $\mu_1$ are identical, we have $\Pr[\mathsf{E}_4] = \Pr[\mathsf{E}_3]$. Moreover, since no information on $\mathsf{coin}$ 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 5.1.** *If* $\mathsf{Com}$ *satisfies perfect binding and* $\Pi_{\mathsf{ZK}}$ *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 $\mathsf{Bad}_b$ 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 $\mathsf{Bad}_0$ or $\mathsf{Bad}_1$ 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 $\Pi_{\mathsf{ZK}}$ 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 $\mathsf{pp}$ 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 $\mathsf{Bad}_0$ occurs, then $\mathcal{P}^*_2$ outputs $(\mathsf{com}, \mathsf{sfe}_{1,0}, \mathsf{sfe}_{2,0})$ and $\mathsf{resp}_0$.

We can see that $\mathcal{P}^*$ perfectly simulates Game 1 for $\mathcal{A}$ until the second stage of $\mathcal{A}$. Moreover, if $\mathsf{Bad}_0$ 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 $\mathsf{Com}$ is perfectly binding. Therefore, by the adaptive soundness with untrusted setup of $\Pi_{\mathsf{ZK}}$, 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 5.1. □

**Lemma 5.2.** *If $\Pi_{\mathsf{SFE}}$ 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 $\Pi_{\mathsf{SFE}}$ 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 $(\sigma_0, \sigma_1)$ to $\mathcal{A}$. Let $\mathsf{coin}'$ 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 $\mathsf{sfe}_1$ 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 $\Pi_{\mathsf{SFE}}$, we have $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\kappa)$. □

Combining the above, Theorem 5.2 is proven. □

**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

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. 2

AO12. Masayuki Abe and Miyako Ohkubo. A framework for universally composable non-committing blind signatures. *Int. J. Appl. Cryptogr.*, 2(3):229–249, 2012. 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. 12

AR05. Dorit Aharonov and Oded Regev. Lattice problems in NP cap conp. *J. ACM*, 52(5):749–765, 2005. 12, 17

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. 2

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. 2

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. 8

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. 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. 2

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. 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. 2

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. 2

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. 2

DN00. Cynthia Dwork and Moni Naor. Zaps and their applications. In *41st FOCS*, pages 283–293. IEEE Computer Society Press, November 2000. 5

DN07. Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007. 5

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. 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. 2

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. 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. 5

FLS99. Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1):1–28, 1999. 5

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. 2

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. 2

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. 2, 5

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. 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. 16

GO94. Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994. 4

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. 2, 3, 4, 5, 6, 13, 15, 22

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. 5

HK12. Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, 25(1):158–193, January 2012. 14

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. 2

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. 3, 5

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

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. 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. 14

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. 4, 5

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. 2

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. 8

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. 11

Reg09. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009. 11, 17

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. 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. 16