Secure Software Leasing from Standard Assumptions

Fuyuki Kitagawa [1] Ryo Nishimaki [1] Takashi Yamakawa [1]

[1] NTT Corporation, Tokyo, Japan
{fuyuki.kitagawa.yh,ryo.nishimaki.zk,takashi.yamakawa.ga}@hco.ntt.co.jp

Abstract

Secure software leasing (SSL) is a quantum cryptographic primitive that enables an authority to lease software to a user by encoding it into a quantum state. SSL prevents users from generating authenticated pirated copies of leased software, where authenticated copies indicate those run on legitimate platforms. Although SSL is a relaxed variant of quantum copy protection that prevents users from generating any copy of leased softwares, it is still meaningful and attractive. Recently, Ananth and La Placa proposed the first SSL scheme. It satisfies a strong security notion called infinite-term security. On the other hand, it has a drawback that it is based on public key quantum money, which is not instantiated with standard cryptographic assumptions so far. Moreover, their scheme only supports a subclass of evasive functions.

In this work, we present SSL schemes that satisfy a security notion called finite-term security based on the learning with errors assumption (LWE). Finite-term security is weaker than infinite-term security, but it still provides a reasonable security guarantee. Specifically, our contributions consist of the following.

- We construct a finite-term secure SSL scheme for pseudorandom functions from the LWE assumption against quantum adversaries.
- We construct a finite-term secure SSL scheme for a subclass of evasive functions from the LWE assumption against sub-exponential quantum adversaries.
- We construct finite-term secure SSL schemes for the functionalities above with classical communication from the LWE assumption against (sub-exponential) quantum adversaries.

SSL with classical communication means that entities exchange only classical information though they run quantum computation locally.

Our crucial tool is two-tier quantum lightning, which is introduced in this work and a relaxed version of quantum lighting. In two-tier quantum lightning schemes, we have a public verification algorithm called semi-verification and a *private* verification algorithm called full-verification. An adversary cannot generate possibly entangled two quantum states whose serial numbers are the same such that one passes the semi-verification, and the other also passes the full-verification. We show that we can construct a two-tier quantum lightning scheme from the LWE assumption.

Keywords: secure software leasing, learning with errors, classical communication

Contents

1 Introduction

1.1 Background

Secure software leasing (SSL) introduced by Ananth and La Placa [AL21] is a quantum cryptographic primitive that enables an authority (the lessor) to lease software[1] to a user (the lessee) by encoding it into a quantum state. SSL prevents users from generating authenticated pirated copies of leased software, where authenticated copies indicate those run on the legitimate platforms.

More specifically, an SSL is the following protocol between the lessor and lessee. The lessor generates a secret key sk used to create a leased version of a circuit C. The leased version is a quantum state and denoted by sft_C. The lessor leases the functionality of C to the lessee by providing sft_C. The lessee can compute $C(x)$ for any input x by using sft_C. That is, there exists a quantum algorithm $\mathcal{Run}$ and it holds that $\mathcal{Run}(\mathit{sft}_C, x) = C(x)$ for any x. The lessor can validate the states returned from the user by using the secret key. That is, there exists a quantum algorithm $\mathcal{Check}$ and $\mathcal{Check}(\mathsf{sk}, \mathit{sft}_C)$ outputs whether sft_C is a valid leased state or not. Since users can create as many copies of classical information as they want, we need the power of quantum computing to achieve SSL.

Ananth and La Placa introduced two security notions for SSL, that is, infinite-term security and finite term security. Infinite-term security guarantees that given a single leased state of a circuit C, adversaries cannot generate possibly entangled bipartite states sft_0^* and sft_1^* both of which can be used to compute C with $\mathcal{Run}$. Finite-term security guarantees that adversaries cannot generate possibly entangled bipartite states sft_0^* and sft_1^* such that $\mathcal{Check}(\mathsf{sk}, \mathit{sft}_0^*) = \top$ (returning a valid leased state) and $\mathcal{Run}(\mathit{sft}_1^*, x) = C(x)$ (adversary still can compute C by using sft_1^*) in an SSL scheme. Roughly speaking, finite-term security guarantees that adversaries cannot compute $C(x)$ via $\mathcal{Run}$ after they return the valid leased state to the lessor.

SSL and copy-protection. Quantum software copy-protection [Aar09] is a closely related notion to SSL. Quantum copy-protection guarantees the following. When adversaries are given a copy-protected circuit for computing C, they cannot create two (possibly entangled) quantum states, both of which can be used to compute C. Here, adversaries are not required to output a quantum state that follows an honest evaluation algorithm $\mathcal{Run}$ (they can use an arbitrary evaluation algorithm $\mathcal{Run}'$). Software copy-protection can be crucial technology to prevent software piracy since users lose software if they re-distribute it. Quantum copy-protection for some circuits class is also known to yield public-key quantum money [ALZ20].

Although SSL is weaker than copy-protection, SSL (with even finite-term security) has useful applications such as limited-time use software, recalling buggy software, preventing drain of propriety software from malicious employees [AL21]. SSL makes software distribution more controllable. In addition, achieving SSL could be a crucial stepping stone to achieve quantum software copy-protection.

One motivative example of (finite-term secure) SSL is a video game platform. A user can borrow a video game title from a company and enjoy it on an appropriate platform (like Xbox of Microsoft). After the user returned the title, s/he cannot enjoy it on the appropriate platform. The title is not guaranteed to work on another (irregular) platform. Thus, SSL is a useful tool in this use case.

(Im)possibility of SSL and copy-protection. Although SSL and software copy-protection have many useful applications, there are few positive results on them. Aaronson observed that learnable functions could not be copy-protected [Aar09]. He also constructed a copy-protection scheme for arbitrary unlearnable Boolean functions relative to a quantum oracle and two *heuristic* copy-protection schemes for point functions in the standard model [Aar09]. Aaronson, Liu, and Zhang constructed a quantum copy-protection scheme for unlearnable functions relative to classical oracles [ALZ20]. There is no secure

[1] Software is modeled as (Boolean) circuits or functions.

quantum copy-protection scheme with a reduction-based proof *without classical/quantum oracles*. We do not know how to implement such oracles under cryptographic assumptions in the previous works.

Ananth and La Placa constructed an infinite-term secure SSL scheme for a sub-class of evasive functions in the common reference string (CRS) model by using public-key quantum money [AC12, Zha21] and the learning with errors (LWE) assumption [AL21]. Evasive functions is a class of functions such that it is hard to find an accepting input (a function outputs 1 for this input) only given black-box access to a function. They also prove that there exists an unlearnable function class such that it is impossible to achieve an SSL scheme for that function class even in the CRS model. The SSL scheme by Ananth and La Placa is the only one positive result without classical/quantum oracles on this topic before our work.[2]

Motivation. There are many fascinating questions about SSL/copy-protection. We focus on the following three questions in this study.

The first one is whether we can achieve SSL/copy-protection from standard assumptions. Avoiding strong assumptions is desirable in cryptography. It is not known whether public-key quantum money is possible under standard assumptions. Zhandry proves that post-quantum indistinguishability obfuscation (IO) [BGI+12] implies public-key quantum money [Zha21]. Several works [CHVW19, AP20, BGMZ18, GP21, BDGM20, WW21] presented candidate constructions of post-quantum secure IO by using lattices.[3] There are several other candidate constructions of public key quantum money [FGH+12, Zha21]. However, none of them has a reduction to standard assumptions.

The second question is whether we can achieve SSL/copy-protection only with classical communication and local quantum computing as in the case of quantum money [RS19, AGKZ20]. Even if quantum computers are available, communicating only classical data is much easier than communicating quantum data over quantum channels. Communication infrastructure might not be updated to support quantum data soon, even after practical quantum computers are commonly used.

The third question is whether we can achieve SSL/copy-protection beyond for evasive functions. The function class is quite limited. For practical software protection, it is crucial to push the function class's boundaries where we can achieve SSL/copy-protection.

1.2 Our Results

We constructed finite-term secure SSL schemes from standard assumptions in this study. We prove the following theorems.

Theorem 1.1 (informal). *Assuming the hardness of the LWE problem against polynomial time quantum adversaries, there is a finite-term secure SSL scheme and SSL scheme with classical communication for pseudorandom functions (PRFs) in the CRS model.*

Theorem 1.2 (informal). *Assuming the hardness of the LWE problem against sub-exponential time quantum adversaries, there is a finite-term secure SSL scheme and SSL scheme with classical communication for a subclass of evasive functions in the CRS model.*

The notable features of our SSL schemes are the following.

- Constructed via a clean and unified framework.
- Secure under standard assumptions (the LWE assumption).

[2] We will refer to a few concurrent works in Section 1.4.

[3] Their constructions need heuristic assumptions related to randomness leakage and circular security [BDGM20, GP21], a heuristic construction of oblivious LWE sampling [WW21], a heuristic construction of noisy linear functional encryption [AP20], or an idealized model [BGMZ18, CHVW19]. Some heuristic assumptions [GP21, WW21, BDGM20] were found to be false [HJL21].

- Can be achieved only with classical communication.
- Supporting functions other than a sub-class of evasive functions.

The crucial tools in our framework are two-tier quantum lighting, which we introduce in this study, and (a relaxed version of) software watermarking [BGI+12, CHN+18]. Two-tier quantum lighting is a weaker variant of quantum lighting [Zha21]. Interestingly, two-tier quantum lightning can be instantiated with standard assumptions, while quantum lightning is not so far. Another exciting feature is that software watermarking can be a building block of SSL. Our study gives a new application of software watermarking. By using these tools, our SSL constructions are modular, and we obtain a clean perspective to achieve SSL. Our abstracted construction ensures that a relaxed watermarking scheme for any circuit class can be converted to SSL for the same class assuming the existence of two-tier QL. As a bonus, our schemes are based on standard assumptions (i.e., do not rely on public-key quantum money). However, our schemes are *finite-term* secure while the scheme by Ananth and La Placa [AL21] is *infinite-term* secure. See Section 1.5 for an overview of our technique, (two-tier) quantum lightning, and software watermarking.

We can achieve SSL schemes with classical communication, where entities send only classical information to other entities (though they generate quantum states for their local computation). Our schemes are the first SSL schemes with classical communication.

We present the first SSL schemes for function classes other than evasive functions. Our schemes open the possibilities of software copy-protection for broader functionalities in the standard model.

1.3 Related Work

Amos, Georgiou, Kiayias, and Zhandry presented many hybrid quantum cryptographic protocols, where we exchange only classical information and local quantum operation can yield advantages [AGKZ20]. Their constructions are secure relative to classical oracles. Radian and Sattath presented the notion of semi-quantum money, where both minting and verification protocols are interactive with classical communication [RS19]. Georgiou and Zhandry presented the notion of unclonable decryption keys [GZ20], which can be seen as quantum copy-protection for specific cryptographic tasks.

1.4 Concurrent Work

Aaronson et al. [ALZ20] significantly revised their paper in October 2020 and added new results in the revised version with additional authors [ALL+21]. They use a similar idea to ours to achieve their additional results. They achieved software copy-detection, which is a version of finite-term secure SSL, from public key quantum money and watermarking. They defined their copy detection so that it can provide natural security guarantee even if we consider leasing decryption or signing functionalities of cryptographic primitives. As previously discussed in the context of watermarking [GKM+19], when considering those functionalities, we need to take a wider class of adversaries into consideration than considering just functions including PRF. In fact, the reason why we focus only on PRF functionalities among cryptographic functionalities is that there was no definition of SSL that can handle decryption or signing functionalities. We believe that by combining the work by Aaronson et al. [ALL+21] and our work, we can realize finite-term secure SSL for decryption and signing functionalities based on the LWE assumption under a reasonable definition.

Coladangelo, Majenz, and Poremba [CMP20] realized finite-term secure SSL for the same sub-class of evasive functions as Ananth and La Placa [AL21] using the quantum random oracle. Based on their work, Broadbent, Jeffery, Lord, Podder, and Sundaram [BJL+21] showed that finite-term secure SSL for the class can be realized without any assumption. We note that the definition of SSL used in these two works is different from the definition by Ananth and La Placa that we basically follow in this work. Their definition has a nice property that their security notion captures any form of pirated copies rather than

just authorized copies. On the other hand, in their definition, not only the security notion, but also the correctness notion is parameterized by distributions on inputs to functions. The security and correctness of the SSL schemes proposed in those works hold with respect to a specific distribution.

The advantage of our results over the above concurrent results is that we achieve SSL for functions beyond evasive functions, that is, PRF under standard lattice assumptions. Moreover, our work is the first one that considers classical communication in the context of SSL.

1.5 Technical Overview

Definition of SSL We review the definition of SSL given in [AL21]. In this paper, we use a calligraphic font to represent quantum algorithms and calligraphic font or bracket notation to represent quantum states following the notation of [AGKZ20].

Formally, an SSL for a function class $\mathcal{C}$ consists of the following algorithms.

$\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$**:** This is a setup algorithm that generates a common reference string.

$\mathsf{Gen}(\mathsf{crs}) \to \mathsf{ssl.sk}$**:** This is an algorithm supposed to be run by the lessor that generates lessor's secret key $\mathsf{ssl.sk}$. The key is used to generate a leased software and verify the validity of a software returned by the lessee.

$\mathcal{L}\mathit{essor}(\mathsf{ssl.sk}, C) \to \mathit{sft}_C$**:** This is an algorithm supposed to be run by the lessor that generates a leased software sft_C that computes a circuit C.

$\mathcal{R}\mathit{un}(\mathsf{crs}, \mathit{sft}_C, x) \to C(x)$**:** This is an algorithm supposed to be run by the lessee to evaluate the software. As correctness, we require that the output should be equal to $C(x)$ with overwhelming probability if sft_C is honestly generated.[4]

$\mathcal{C}\mathit{heck}(\mathsf{ssl.sk}, \mathit{sft}_C) \to \top/\bot$**:** This is an algorithm supposed to be run by the lessor to check the validity of the software sft_C returned by the lessee. As correctness, we require that this algorithm returns $\top$ (i.e., it accepts) with overwhelming probability if sft_C is an honestly generated one.

In this work, we focus on finite-term secure SSL. Roughly speaking, the finite-term security of SSL requires that no quantum polynomial time (QPT) adversary given sft_C (for randomly chosen C according to a certain distribution) can generate (possibly entangled) quantum states sft_0 and sft_1 such that $\mathcal{C}\mathit{heck}(\mathsf{ssl.sk}, \mathit{sft}_0) \to \top$ and $\mathcal{R}\mathit{un}(\mathsf{crs}, \mathit{sft}_1, \cdot)$ computes C with non-negligible probability. Thus, intuitively, the finite-term security ensures that finite-term security guarantees that adversaries cannot compute $C(x)$ via $\mathcal{R}\mathit{un}$ after they return the valid leased state to the lessor.

Construction of SSL in [AL21] We review the construction of SSL in [AL21]. Their construction is based on the following three building blocks:

Publicly verifiable unclonable state generator. This enables us to generate a pair $(\mathsf{pk}, \mathsf{sk})$ of public and secret keys in such a way that the following conditions are satisfied:

1. Given sk, we can efficiently generate a quantum state $|\psi_{\mathsf{pk}}\rangle$.
2. Given pk, we can efficiently implement a projective measurement $\{|\psi_{\mathsf{pk}}\rangle\langle\psi_{\mathsf{pk}}|, I - |\psi_{\mathsf{pk}}\rangle\langle\psi_{\mathsf{pk}}|\}$.
3. Given pk and $|\psi_{\mathsf{pk}}\rangle$, no QPT algorithm can generate $|\psi_{\mathsf{pk}}\rangle^{\otimes 2}$ with non-negligible probability.

Aaronson and Christiano [AC12] constructed a publicly verifiable unclonable state generator (under the name "quantum money mini-scheme") relative to a classical oracle, and Zhandry [Zha21] gave an instantiation in the standard model assuming post-quantum IO.

[4] In the actual syntax, it also outputs a software, which is negligibly close to a software given as input.

Input-hiding obfuscator. This converts a circuit $C \in \mathcal{C}$ (that is taken from a certain distribution) to a functionally equivalent obfuscated circuit $\widetilde{C}$ in such a way that no QPT algorithm given $\widetilde{C}$ can find accepting point i.e., x such that $C(x) = 1$.

Ananth and La Placa [AL21] constructed an input-hiding obfuscator for a function class called compute-and-compare circuits under the LWE assumption.[5]

Simulation-extractable non-interactive zero-knowledge. A non-interactive zero-knowledge (NIZK) enables a prover to non-interactively prove an NP statement without revealing anything beyond the truth of the statement assuming a common reference string (CRS) generated by a trusted third party. A simulation-extractable NIZK (seNIZK) additionally enables us to extract a witness from an adversary that is given arbitrarily many proofs generated by a zero-knowledge simulator and generates a new valid proof. This property especially ensures that an seNIZK is an *argument of knowledge* where a prover can prove not only truth of a statement but also that it knows a witness for the statement.

Ananth and La Placa [AL21] showed that an seNIZK can be constructed from any (non-simulation-extractable) NIZK and CCA secure PKE, which can be instantiated under the LWE assumption [PS19, PW11].

Then their construction of SSL for $\mathcal{C}$ is described as follows:

$\mathsf{Setup}(1^\lambda)$**:** This just generates and outputs a CRS crs of seNIZK.

$\mathsf{Gen}(\mathsf{crs})$**:** This generates a pair $(\mathsf{pk}, \mathsf{sk})$ of public and secret keys of the publicly verifiable unclonable state generator and outputs $\mathsf{ssl.sk} := (\mathsf{pk}, \mathsf{sk})$.

$\mathcal{L}\mathit{essor}(\mathsf{ssl.sk} = (\mathsf{pk}, \mathsf{sk}), C)$**:** This obfuscates C to generate an obfuscated circuit $\widetilde{C}$ by the input-hiding obfuscator and generates an seNIZK proof π for a statement $(\mathsf{pk}, \widetilde{C})$ that it knows an accepting input x of $\widetilde{C}$.[6] Then it outputs a leased software $\mathit{sft}_C := (|\psi_{\mathsf{pk}}\rangle, \mathsf{pk}, \widetilde{C}, \pi)$. We call $|\psi_{\mathsf{pk}}\rangle$ and $(\mathsf{pk}, \widetilde{C}, \pi)$ as quantum and classical parts of sft_C, respectively.

$\mathcal{R}\mathit{un}(\mathsf{crs}, \mathit{sft}_C, x)$**:** This immediately returns $\bot$ if π does not pass the verification of seNIZK. It performs a projective measurement $\{|\psi_{\mathsf{pk}}\rangle\langle\psi_{\mathsf{pk}}|, I - |\psi_{\mathsf{pk}}\rangle\langle\psi_{\mathsf{pk}}|\}$ on the quantum part of sft_C by using pk and if the latter projection was applied, then it returns $\bot$. Otherwise, it outputs $\widetilde{C}(x)$.

$\mathcal{C}\mathit{heck}(\mathsf{ssl.sk}, \mathit{sft}_C)$**:** It performs a projective measurement $\{|\psi_{\mathsf{pk}}\rangle\langle\psi_{\mathsf{pk}}|, I - |\psi_{\mathsf{pk}}\rangle\langle\psi_{\mathsf{pk}}|\}$ on the quantum part of sft_C and returns $\top$ if the former projection was applied and $\bot$ otherwise.

Intuitively, the finite-term security of the above SSL can be proven as follows.[7] Suppose that there exists an adversary that is given $\mathit{sft}_C = (|\psi_{\mathsf{pk}}\rangle, \mathsf{pk}, \widetilde{C}, \pi)$ and generates $\mathit{sft}_0 = (\mathit{psi}_0, \mathsf{pk}_0, \widetilde{C}_0, \pi_0)$ and $\mathit{sft}_1 = (\mathit{psi}_0, \mathsf{pk}_1, \widetilde{C}_1, \pi_1)$ such that $\mathcal{C}\mathit{heck}(\mathsf{ssl.sk}, \mathit{sft}_0) \to \top$ and $\mathcal{R}\mathit{un}(\mathsf{crs}, \mathit{sft}_1, \cdot)$ computes C with non-negligible probability. Then we consider the following two cases:

Case 1. $\mathsf{pk}_1 = \mathsf{pk}$: In this case, if $\mathcal{R}\mathit{un}(\mathsf{crs}, \mathit{sft}_1, \cdot)$ correctly computes C (and especially outputs a non-$\bot$ value), then the quantum part of sft_1 after the execution should be $|\psi_{\mathsf{pk}}\rangle$ by the construction of

[5] A compute-and-compare circuit is specified by a circuit C and a target value α and outputs 1 on input x if and only if $C(x) = \alpha$.

[6] In the original construction in [AL21], seNIZK also proves that pk and $\widetilde{C}$ was honestly generated. However, we found that this is redundant, and essentially the same security proof works even if it only proves the knowledge of an accepting input of $\widetilde{C}$. We note that it is important to include pk in the statement to bind a proof to pk even though the knowledge proven by the seNIZK has nothing to do with pk. In fact, this observation is essential to give our simplified construction of SSL.

[7] Note that Ananth and La Placa proved that the construction in fact satisfies infinite-term security that is stronger than finite-term security. For ease of exposition of our ideas, we explain why the construction satisfies finite-term security.

$\mathcal{Run}$. On the other hand, if we have $\mathcal{Check}(\mathsf{ssl.sk}, \mathit{sft}_0) \to \top$, then the quantum part of sft_0 after the verification should also be $|\psi_{\mathsf{pk}}\rangle$ by the definition of the verification. Therefore, they can happen simultaneously only with a negligible probability due to the unclonability of $|\psi_{\mathsf{pk}}\rangle$.

Case 2. $\mathsf{pk}_1 \neq \mathsf{pk}$: In this case, if $\mathcal{Run}(\mathsf{crs}, \mathit{sft}_1, \cdot)$ correctly computes C, then π_1 is a valid proof for a statement $(\mathsf{pk}_1, \widetilde{C}_1)$ and $\widetilde{C}_1$ is functionally equivalent to C. Since we have $(\mathsf{pk}_1, \widetilde{C}_1) \neq (\mathsf{pk}, \widetilde{C})$, by the simulation extractability of seNIZK, even if we replace π with a simulated proof, we can extract a witness for $(\mathsf{pk}_1, \widetilde{C}_1)$, which contains an accepting input for C. Since simulation of π can be done only from the statement $(\mathsf{pk}, \widetilde{C})$, this contradicts security of the input-hiding obfuscator, and thus this happens with a negligible probability.

In summary, an adversary cannot win with a non-negligible probability in either case, which means that the SSL is finite-term secure.

Our idea for weakening assumptions. Unfortunately, their construction is based on a very strong assumption of post-quantum IO, which is needed to construct a publicly verifiable unclonable state generator. Indeed, a publicly verifiable unclonable state generator implies public key quantum money by combining it with digital signatures [AC12]. Therefore, constructing a publicly verifiable unclonable state generator is as difficult as constructing a public key quantum money scheme, which is not known to exist under standard assumptions.

Our main observation is that we actually do not need the full power of public key quantum money for the above construction of SSL if we require only finite-term security since $\mathcal{Check}$ can take a secret key, and thus it can run a private verification algorithm. Then, does secret key quantum money suffice? Unfortunately, the answer is no. The reason is that even though $\mathcal{Check}$ can take a secret key, $\mathcal{Run}$ cannot since the secret key should be hidden from the lessee. Based on this observation, we can see that what we actually need is something between public key quantum money and secret key quantum money. We formalize this as *two-tier quantum lightning*, which is a significant relaxation of quantum lightning introduced by Zhandry [Zha21].

Two-tier quantum lightning. Roughly speaking, quantum lightning (QL) is a special type of public key quantum money where anyone can generate a money state. In QL, a public key pk is published by a setup algorithm and given pk, anyone can efficiently generate a serial number snum along with a corresponding quantum state called bolt, which we denote by bolt. We call this a *bolt generation* algorithm. As correctness, we require that given pk, snum, and any quantum state bolt, anyone can verify if bolt is a valid state corresponding to the serial number snum. Especially, if bolt is an honestly generated bolt, then the verification accepts with overwhelming probability. On the other hand, as security, we require that no QPT algorithm given pk can generate two (possibly entangled) quantum states bolt_0 and bolt_1 and a serial number snum such that both states pass the verification w.r.t. the serial number snum with non-negligible probability.

We introduce a weaker variant of QL which we call *two-tier QL*. In two-tier QL, a setup algorithm generates both a public key pk and a secret key sk, and given pk, anyone can efficiently generate a serial number snum along with a corresponding quantum state bolt similarly to the original quantum lightning. The main difference from the original QL is that it has two types of verification: *full-verification* and *semi-verification*. Full-verification uses a secret key sk while semi-verification only uses a public key pk. As correctness, we require that an honestly generated bolt passes both verifications with overwhelming probability. On the other hand, as security, we require that no QPT algorithm given pk can generate two (possibly entangled) quantum states bolt_0 and bolt_1 and a serial number snum such that bolt_0 passes the *full-verification* w.r.t. the serial number snum and bolt_1 passes the *semi-verification* w.r.t. the serial number snum with non-negligible probability. We note that this does not prevent an adversary from

generating two states that pass semi-verification. Thus, we cannot use the semi-verification algorithm as a verification algorithm of the original QL.

We show that this two-tier verification mechanism is a perfect fit for finite-term secure SSL. Specifically, based on the observation that $\mathcal{Check}$ can take a secret key whereas $\mathcal{Run}$ cannot as explained in the previous paragraph, we can use two-tier QL instead of publicly verifiable quantum state generators. This replacement is a slight adaptation of the construction in [AL21] by implementing verification by $\mathcal{Check}$ and $\mathcal{Run}$ with full- and semi-verification of two-tier QL, respectively. We omit the detailed construction since that is mostly the same as that in [AL21] except that we use two-tier QL.

Constructions of two-tier quantum lightning. Although no known construction of the original QL is based on a standard assumption, we give two two-tier QL schemes based on standard assumptions.

The first construction is based on the SIS assumption inspired by the recent work by Roberts and Zhandry [RZ21]. The SIS assumption requires that no QPT algorithm given a matrix $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times m}$ can find a short $\boldsymbol{s} \in \mathbb{Z}^m$ such that $\boldsymbol{A}\boldsymbol{s} = 0 \mod q$. Using this assumption, a natural approach to construct QL is as follows:[8] Given a public key $\boldsymbol{A}$, a bolt generation algorithm generates a bolt of the form $\sum_{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{y} \text{ and } \boldsymbol{x} \text{ is ``short''}} \alpha_{\boldsymbol{x}} |\boldsymbol{x}\rangle$ and a corresponding serial number $\boldsymbol{y}$. This can be done by generating a superposition of short vectors in $\mathbb{Z}^m$, multiplying by $\boldsymbol{A}$ in superposition to write the result in an additional register, and measuring it. The SIS assumption ensures that no QPT algorithm can generate two copies of a well-formed bolt for the same serial number with non-negligible probability. If it is possible, one can break the SIS assumption by measuring both bolts and returns the difference between them as a solution. However, the fundamental problem is that we do not know how to publicly verify that a given state is a well-formed bolt for a given serial number. Roughly speaking, Roberts and Zhandry showed that such verification is possible given a trapdoor behind the matrix $\boldsymbol{A}$, which yields a secretly verifiable version of QL (which is formalized as *franchised quantum money* in [RZ21]). We use this verification as the full-verification of our two-tier QL. On the other hand, we define a semi-verification algorithm as an algorithm that just checks that a given state is a superposition of short preimages of $\mathsf{snum} = \boldsymbol{y}$ regardless of whether it is a well-formed superposition or not. This can be done by multiplying $\boldsymbol{A}$ in superposition, and especially can be done publicly. Though a state that passes the semi-verification may collapse to a classical state, a state that passes the full-verification should not. Therefore, if we measure states that pass full- and semi- verification w.r.t. the same serial number, then the measurement outcomes are different with non-negligible probability. Thus the difference between them gives a solution to the SIS problem. This implies that this construction of two-tier QL satisfies the security assuming the SIS assumption.

The second construction is based on the LWE assumption. The design strategy is based on a similar idea to the proof of quantumness by Brakerski et al. [BCM+18]. We especially use a family of noisy trapdoor claw-free permutations constructed based on the LWE assumption in [BCM+18]. For simplicity, we describe the construction based on a family of clean (non-noisy) trapdoor claw-free permutations in this overview. A family of trapdoor claw-free permutations enables us to generate a function $f : \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ such that both $f(0,\cdot)$ and $f(1,\cdot)$ are permutations along with a trapdoor. As claw-free property, we require that no QPT algorithm given a description of f can generate $x_0, x_1 \in \{0,1\}^n$ such that $f(0,x_0) = f(1,x_1)$ with non-negligible probability. On the other hand, if one is given a trapdoor, then one can efficiently computes x_0, x_1 such that $f(0,x_0) = f(1,x_1) = y$ for any $y \in \{0,1\}^n$. Based on this, we construct two-tier QL as follows: The setup algorithm generates f and its trapdoor td, and sets a public key as the function f and secret key as the trapdoor td. A bolt generation algorithm first prepares a uniform superposition $\sum_{b\in\{0,1\},x\in\{0,1\}^n} |b\rangle |x\rangle$, applies f in superposition to generate $\sum_{b\in\{0,1\},x\in\{0,1\}^n} |b\rangle |x\rangle |f(b,x)\rangle$, measures the third register to obtain $y \in \{0,1\}^n$ along with a collapsed state $\frac{1}{\sqrt{2}}(|0\rangle |x_0\rangle + |1\rangle |x_1\rangle)$ where $f(0,x_0) = f(1,x_1) = y$. Then it outputs a serial number $\mathsf{snum} := y$ and a bolt $\mathit{bolt} := \frac{1}{\sqrt{2}}(|0\rangle |x_0\rangle + |1\rangle |x_1\rangle)$. The full-

[8] This approach was also discussed in the introduction of [Zha21].

verification algorithm given a trapdoor td, a serial number $\mathsf{snum} = y$, and a (possibly malformed) bolt bolt, computes x_0, x_1 such that $f(0,x_0) = f(1,x_1) = y$ using the trapdoor, and checks if bolt is $\frac{1}{\sqrt{2}}(|0\rangle|x_0\rangle + |1\rangle|x_1\rangle)$. More formally, it performs a projective measurement $\{\Pi, I-\Pi\}$ where $\Pi := \frac{1}{2}(|0\rangle|x_0\rangle + |1\rangle|x_1\rangle)(\langle 0|\langle x_0| + \langle 1|\langle x_1|)$ and accepts if Π is applied. The semi-verification algorithm given f, $\mathsf{snum} = y$ and a (possibly malformed) bolt bolt just checks that bolt is a (not necessarily uniform) superposition of $(0,x_0)$ and $(1,x_1)$ by applying f in superposition. Suppose that we are given states bolt_0 and bolt_1 that pass the full- and semi-verification respectively w.r.t. the same serial number $\mathsf{snum} = y$. Then after these verifications accept, if we measure bolt_0, then we get x_0 or x_1 with equal probability and if we measure bolt_1, we get either of x_0 and x_1. Therefore, with probability $1/2$, we obtain both x_0 and x_1, which contradicts the claw-free property. Thus, the above two-tier QL is secure under the claw-free property.

Abstracted construction of SSL via watermarking. Besides weakening the required assumption, we also give a slightly more abstracted SSL construction through the lens of watermarking. In general, a watermarking scheme enables us to embed a mark into a program so that the mark cannot be removed or modified without significantly changing the functionality. We observe that the classical part $(\mathsf{pk}, \widetilde{C}, \pi)$ of a leased software of [AL21] can be seen as a watermarked program of C where pk is regarded as a mark. In this context, we only need to ensure that one cannot remove or modify the mark as long as one does not change the program's functionality *when it is run on a legitimate evaluation algorithm* similarly to the security requirement for SSL. We call a watermarking with such a weaker security guarantee a *relaxed watermarking*. With this abstraction along with the observation that two-tier QL suffices as already explained, we give a generic construction of SSL for $\mathcal{C}$ based on two-tier QL and relaxed watermarking for $\mathcal{C}$. This construction is in our eyes simpler than that in [AL21].[9] From this point of view, we can see that [AL21] essentially constructed a relaxed watermarking for compute-and-compare circuits based on seNIZK and input-hiding obfuscator for compute-and-compare circuits. We observe that an input-hiding obfuscator for compute-and-compare circuits can be instantiated from any injective one-way function, which yields a simpler construction of relaxed watermarking for compute-and-compare circuits without explicitly using input-hiding obfuscators.

SSL for PRF. Our abstracted construction ensures that a relaxed watermarking scheme for any circuit class can be converted to SSL for the same class assuming the existence of two-tier QL. Here, we sketch our construction of a relaxed watermarking scheme for PRF. Let F_K be a function that evaluates a PRF with a key K. We assume that the PRF is a puncturable PRF. That is, one can generate a punctured key K_{x^*} for any input x^* that can be used to evaluate F_K on all inputs except for x^* but $F_K(x^*)$ remains pseudorandom even given K_{x^*}. For generating a watermarked version of F_K with a mark m, we generate $(K_{x^*}, y^* := F_K(x^*))$ for any fixed input x^* and an seNIZK proof π for a statement $(\mathsf{m}, K_{x^*}, y^*)$ that it knows K. Then a watermarked program is set to be $(\mathsf{m}, K_{x^*}, y^*, \pi)$. A legitimate evaluation algorithm first checks if π is a valid proof, and if so evaluates F_K by using K_{x^*} and y^*, and returns $\bot$ otherwise. Roughly speaking, this construction satisfies the security of relaxed watermarking since if an adversary is given $(\mathsf{m}, K_{x^*}, y^*, \pi)$ can generate a program with a mark $\mathsf{m}' \neq \mathsf{m}$ that correctly computes F_K on the legitimate evaluation algorithm. The program should contain a new valid proof of seNIZK that is different from π. By the simulation extractability, we can extract K by using such an adversary. Especially, this enables us to compute K from (K_{x^*}, y^*), which contradicts security of the puncturable PRF. [10]

By plugging the above relaxed watermarking for PRF into our generic construction, we obtain SSL for PRF. This would be impossible through the abstraction of [AL21] since input-hiding obfuscator can exist only for evasive functions, whereas PRF is not evasive.

[9] Strictly speaking, our construction additionally uses message authentication code (MAC).

[10] Strictly speaking, we need to assume the key-injectiveness for the PRF as defined in Definition 2.8.

SSL with classical communication. As a final contribution, we give a construction of finite-term secure SSL where communication between the lessor and lessee is entirely classical. At a high level, the only quantum component of our SSL is two-tier QL, which can be seen as a type of quantum money. Thus we rely on techniques used for constructing semi-quantum money [RS19], which is a (secret key) quantum money with classical communication. More details are explained below.

In the usage scenario of finite-term secure SSL, there are two parts where the lessor and lessee communicate through a quantum channel. The first is when the lessor sends a software to the lessee. The second is when the lessee returns the software to the lessor.

For removing the first quantum communication, we observe that the only quantum part of a software is a bolt of two-tier QL in our construction, which can be generated publicly. Then, our idea is to let the lessee generate the bolt by himself and only send the corresponding serial number to ask the lessor to generate a classical part of a software while keeping the bolt on lessee's side. This removes the quantum communication at the cost of introducing an interaction. Though we let the lessor generate a bolt and a serial number by himself, the security of SSL is not affected because the security of two-tier QL ensures that an adversary cannot clone a bolt even if it is generated by himself.

For removing the second quantum communication, we assume an additional property for two-tier QL called *bolt-to-certificate capability*, which was originally considered for (original) QL [CS20]. Intuitively, this property enables us to convert a bolt to a classical certificate that certifies that the bolt was broken. Moreover, it certifies that one cannot generate any state that passes the semi-verification. With this property, when returning the software, instead of sending the software itself, it can convert the bolt to a corresponding certificate and then send the classical certificate. Security is still maintained with this modification since if the verification of the certification passes, then this ensures that the lessee no longer possesses a state that passes the semi-verification, and thus $\mathcal{Run}$ always returns $\bot$.

Finally, we show that our LWE-based two-tier QL can be modified to have the bolt-to-certificate capability based on ideas taken from [BCM+18, RS19]. Recall that in the LWE-based construction, a bolt is of the form $\frac{1}{\sqrt{2}}(|0\rangle|x_0\rangle + |1\rangle|x_1\rangle)$. If we apply a Hadamard transform to the state and then measures both registers in the standard basis, then we obtain (m,d) such that $m = d\cdot(x_0\oplus x_1)$ as shown in [BCM+18]. Moreover, Brakerski et al. [BCM+18] showed that the LWE-based trapdoor claw-free permutation satisfies a nice property called *adaptive hardcore property*, which roughly means that no QPT algorithm can output (m,d,x',y) such that $d\neq 0$, $m = d\cdot(x_0\oplus x_1)$ and $x'\in\{x_0,x_1\}$ with probability larger than $1/2+\mathsf{negl}(\lambda)$ where x_0 and x_1 are the unique values such that $f(0,x_0) = f(1,x_1) = y$.[11] Since a quantum state that passes the semi-verification w.r.t. a serial number y is a (not necessarily uniform) superposition of x_0 and x_1, we can see that (m,d) works as a certificate with a weaker security guarantee that if one keeps a quantum state that passes the semi-verification, then one can generate (m,d) that passes verification of $m = d\cdot(x_0\oplus x_1)$ with probability at most $1/2+\mathsf{negl}(\lambda)$. But this still does not suffice for our purpose since one can generate a certificate that passes the verification without discarding the original bolt with probability $1/2$ by just randomly guessing (m,d). To reduce this probability to negligible, we rely on an amplification theorem in [RS19] (which in turn is based on [CHS05]). As a result, we can show that a parallel repetition to the above construction yields a two-tier QL with the bolt-to-certificate capability.

2 Preliminaries

We review notations and definitions of cryptographic tools used in this paper.

[11] More precisely, they prove an analogous property for a family of noisy trapdoor claw-free permutations.

2.1 Notations

In this paper, standard math or sans serif font stands for classical algorithms (e.g., C or Gen) and classical variables (e.g., x or pk). Calligraphic font stands for quantum algorithms (e.g., $\mathcal{Gen}$) and calligraphic font and/or the bracket notation for (mixed) quantum states (e.g., $\mathcal{sk}$ or $|\psi\rangle$).

In this paper, for a finite set X and a distribution D, $x \leftarrow X$ denotes selecting an element from X uniformly at random, $x \leftarrow D$ denotes sampling an element x according to D, and Let $y \leftarrow \mathsf{A}(x)$ and $y \leftarrow \mathcal{A}(\chi)$ denote assigning to y the output of a probabilistic or deterministic algorithm A and a quantum algorithm $\mathcal{A}$ on an input x and χ, respectively. When we explicitly show that A uses randomness r, we write $y \leftarrow \mathsf{A}(x;r)$. Let $[\ell]$ denote the set of integers $\{1,\cdots,\ell\}$, λ denote a security parameter, and $y := z$ denote that y is set, defined, or substituted by z. PPT and QPT algorithms stand for probabilistic polynomial time algorithms and polynomial time quantum algorithms, respectively. Let negl denote a negligible function.

Let X be a random variable over a set S. The *min-entropy* of X, denoted by $\mathrm{H}_\infty(X)$, is defined by $\mathrm{H}_\infty(X) := -\log_2 \max_{x\in S} \Pr[X=x]$. The conditional min-entropy of X conditioned on a correlated variable Y, denoted by $\mathrm{H}_\infty(X|Y)$, is defined as $\mathrm{H}_\infty(X|Y) := -\log_2 \left(\mathrm{E}_{y\leftarrow Y}\left[\max_{x\in S} \Pr[X=x|Y=y]\right]\right)$.

Let $\mathcal{H}$ denote a finite-dimensional Hilbert space. For an operator X on $\mathcal{H}$, let $\|X\|$ denote the operator norm of X, and $\|X\|_{\mathrm{tr}} := \frac{1}{2}\|X\|_1 = \frac{1}{2}\sqrt{XX^\dagger}$ for the trace norm.

2.2 Distributions and Distance

- D: a distribution over a finite domain X.
- f: density on X. That is, a function $f : X \to [0,1]$ such that $\sum_{x\in X} f(x) = 1$.
- $\mathcal{D}_X$: the set of all densities on X.
- For any $f \in \mathcal{D}_X$, $\mathsf{Supp}(f) := \{x \in X \mid f(x) > 0\}$.
- For two densities f_0 and f_1 over the same finite domain X, the Hellinger distance between f_0 and f_1 is
$$\mathsf{H}^2(f_0,f_1) := 1 - \sum_{x\in X} \sqrt{f_0(x)f_1(x)}.$$
- For density matrices $\mathcal{X},\mathcal{Y}$, the trace distance $\|\mathcal{X}-\mathcal{Y}\|_{\mathrm{tr}}$ is equal to
$$\frac{1}{2}\mathrm{Tr}(\sqrt{(\mathcal{X}-\mathcal{Y})^2}).$$

The following lemma relates Hellinger distance and the trace distance of superpositions.

Lemma 2.1. *Let X be a finite set, $f_0, f_1 \in \mathcal{D}_X$, and*
$$|\psi_b\rangle := \sum_{x\in X} \sqrt{f_b(x)}\,|x\rangle$$
for $b \in \{0,1\}$. It holds that
$$\||\psi_0\rangle\langle\psi_0| - |\psi_1\rangle\langle\psi_1|\|_{\mathrm{tr}} = \sqrt{1-(1-\mathsf{H}^2(f_0,f_1))^2}.$$

2.3 Lattices

Definition 2.2 (Learning with Errors). *Let $n, m, q \in \mathbb{N}$ be integer functions of the security parameter λ. Let $\chi = \chi(\lambda)$ be a error distribution over $\mathbb{Z}$. The LWE problem $\mathrm{LWE}_{n,m,q,\chi}$ is to distinguish the following two distributions.*

$$D_0 := \{(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}) \mid \boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times m}, \boldsymbol{s} \leftarrow \mathbb{Z}_q^n, \boldsymbol{e} \leftarrow \chi^m\} \text{ and } D_1 := \{(\boldsymbol{A}, \boldsymbol{u}) \mid \boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times m}, \boldsymbol{u} \leftarrow \mathbb{Z}_q^m\}.$$

When we say we assume the quantum hardness of the LWE problem, we assume that for any QPT adversary $\mathcal{A}$, it holds that

$$|\Pr[\mathcal{A}(D_0) = 1] - \Pr[\mathcal{A}(D_1) = 1]| \leq \mathsf{negl}(\lambda).$$

Definition 2.3 (Short Integer Solution). *Let $n, m, q \in \mathbb{N}$ be integer functions of the security parameter λ. The SIS problem $\mathrm{SIS}_{n,m,q,\beta}$ is as follows. Given $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times m}$ and a positive real β, find a non-zero vector $\boldsymbol{s} \in \mathbb{Z}^m$ such that $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{0} \bmod q$ and $\|s\| \leq \beta$.*

When we say we assume the quantum hardness of the SIS problem, we assume that for any QPT adversary $\mathcal{A}$, it holds that

$$\Pr\Big[\boldsymbol{A}\boldsymbol{s} = \boldsymbol{0} \bmod q \wedge \|\boldsymbol{s}\| \leq \beta \mid \boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times m}, \boldsymbol{s} \leftarrow \mathcal{A}(\boldsymbol{A}, \beta)\Big] \leq \mathsf{negl}(\lambda).$$

2.4 One-Way Functions

We introduce the definition of a family of one-way functions (OWF) for high min-entropy sources.

Definition 2.4 (OWF for High Min-Entropy Sources). *Let $\mathcal{F}_{\mathsf{ow}} = \{f : D_{\mathsf{owf}} \to R_{\mathsf{owf}}\}$ be a family of efficiently computable deterministic functions. Let $\gamma(\lambda)$ be a function and $\mathcal{D}$ a distribution $\mathcal{D} = \{\mathcal{D}_\lambda\}_\lambda$, where $(x, \mathsf{z}) \leftarrow \mathcal{D}_\lambda$ outputs $x \in D_{\mathsf{owf}}$ and some auxiliary information z such that $\mathrm{H}_\infty(x|\mathsf{z}) \geq \alpha(\lambda)$. We say that $\mathcal{F}_{\mathsf{ow}}$ is a family of OWF for for α-sources if for all QPT adversaries $\mathcal{A}$, we have*

$$\Pr\left[f(x') = y \;\middle|\; \begin{array}{c} f \leftarrow \mathcal{F}_{\mathsf{ow}} \\ (x, \mathsf{z}) \leftarrow \mathcal{D}_\lambda \\ x' \leftarrow \mathcal{A}(1^\lambda, f, \mathsf{z}, f(x)) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Alwen, Krenn, Pietrzak, and Wichs [AKPW13] prove that we can achieve deterministic encryption secure for any λ^η min-entropy source for any $\eta > 0$ under the LWE assumption. Such deterministic encryption implies a family of injective OWF for λ^η-sources. This is the case when we consider QPT adversaries. Formally, we have the following theorem.

Theorem 2.5. *Let $\eta > 0$ be any constant. Assuming the quantum hardness of the LWE problem, there exists a post-quantum injective OWF family for λ^η-sources.*

2.5 Pseudorandom Functions and Related Notions

We introduce the definitions of pseudorandom functions (PRF) and puncturable PRF.

Definition 2.6 (Pseudorandom Functions). *For sets $\mathcal{K}$, $\{0,1\}^n$, and $\{0,1\}^m$, let $\{\mathsf{F}_\mathsf{K}(\cdot) : \{0,1\}^n \to \{0,1\}^m \mid \mathsf{K} \in \mathcal{K}\}$ be a family of polynomially computable functions. We say that F is pseudorandom if for any QPT adversary $\mathcal{A}$, it holds that*

$$\begin{aligned} \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{F},\mathcal{A}}(\lambda) \;=\; & \big|\Pr\Big[\mathcal{A}^{\mathsf{F}_\mathsf{K}(\cdot)}(1^\lambda) = 1 : \mathsf{K} \leftarrow \mathcal{K}\Big] \\ & - \Pr\Big[\mathcal{A}^{\mathsf{R}(\cdot)}(1^\lambda) = 1 : \mathsf{R} \leftarrow \mathcal{U}\Big]\big| = \mathsf{negl}(\lambda)\ , \end{aligned}$$

where $\mathcal{U}$ is the set of all functions from $\{0,1\}^n$ to $\{0,1\}^m$.

Definition 2.7 (Puncturable PRF). *For sets $\{0,1\}^n$ and $\{0,1\}^m$, a puncturable PRF* PPRF *whose key space is $\mathcal{K}$ consists of a tuple of algorithms* $(\mathsf{PRF.Eval}, \mathsf{Puncture}, \mathsf{PRF.pEval})$ *that satisfies the following two conditions.*

Functionality preserving under puncturing: *For all polynomial size subset $\{x_i\}_{i\in[k]}$ of $\{0,1\}^n$, all $x \in \{0,1\}^n \setminus \{x_i\}_{i\in[k]}$, and all $\mathsf{K} \in \mathcal{K}$, we have $\mathsf{PRF.Eval}(\mathsf{K}, x) = \mathsf{PRF.pEval}(\mathsf{K}^*, x)$, where $\mathsf{K}^* \leftarrow \mathsf{Puncture}(\mathsf{K}, \{x_i\}_{i\in[k]})$.*

Pseudorandomness at punctured points: *For all polynomial size subset $\{x_i\}_{i\in[k]}$ of $\{0,1\}^n$, and any QPT adversary $\mathcal{A}$, it holds that*

$$\left|\Pr\left[\mathcal{A}(\mathsf{K}^*, \{\mathsf{PRF.Eval}(\mathsf{K}, x_i)\}_{i\in[k]}) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{K}^*, \mathcal{U}^k) = 1\right]\right| = \mathsf{negl}(\lambda) \ ,$$

where $\mathsf{K} \leftarrow \mathcal{K}$, $\mathcal{K}^* \leftarrow \mathsf{Puncture}(\mathsf{K}, \{x_i\}_{i\in[k]})$, and $\mathcal{U}$ denotes the uniform distribution over $\{0,1\}^m$.

We recall the notion of key-injectiveness for puncturable PRF [CHN+18].

Definition 2.8 (Key-Injectiveness). *We say that a puncturable PRF* PPRF *is key-injective if we have*

$$\Pr_{\mathsf{K}\leftarrow\mathcal{K}}[\exists x \in \{0,1\}^n, \mathsf{K}' \in \mathcal{K} \text{ s.t. } \mathsf{K} \neq \mathsf{K}' \wedge \mathsf{PRF.Eval}(\mathsf{K}, x) = \mathsf{PRF.Eval}(\mathsf{K}', x)] \leq \mathsf{negl}(\lambda).$$

We can realize puncturable PRF based on any one-way function. Moreover, even if we require key-injectiveness, we can realize it under the LWE assumption, as shown by Cohen et al. [CHN+18]. This is the case when we consider QPT adversaries. Formally, we have the following theorem.

Theorem 2.9. *There exists a key-injective puncturable PRF secure against QPT adversaries assuming the quantum hardness of the LWE problem.*

2.6 One-Time Message Authentication Code

We introduce the definition of one-time message authentication code (OT-MAC).

Definition 2.10 (OT-MAC). *An OT-MAC* MAC *is a three tuple* $(\mathsf{MAC.Gen}, \mathsf{MAC.Tag}, \mathsf{MAC.Vrfy})$ *of PPT algorithms. Below, let D_{mac} be the domain of* MAC.

- $\mathsf{MAC.Gen}(1^\lambda)$: *Given a security parameter 1^λ, outputs a key* s.
- $\mathsf{MAC.Tag}(\mathsf{s}, m)$: *Given a key* s *and a message $m \in D_{\mathsf{mac}}$, outputs* tag.
- $\mathsf{MAC.Vrfy}(\mathsf{s}, m, \mathsf{tag})$: *Given a key* s*, message $m \in D_{\mathsf{mac}}$, and* tag*, outputs $\top$ or $\bot$.*

We require the following properties.

Correctness: *For every $m \in D_{\mathsf{mac}}$ and $\mathsf{s} \leftarrow \mathsf{MAC.Gen}(1^\lambda)$, we have $\mathsf{MAC.Vrfy}(\mathsf{s}, m, \mathsf{MAC.Tag}(\mathsf{s}, m)) = \top$.*

Security: *For any QPT adversary $\mathcal{A}$, it holds that*

$$\Pr\left[\begin{array}{l|l} \mathsf{MAC.Vrfy}(\mathsf{s}, m, \mathsf{tag}) = \top \wedge & \mathsf{s} \leftarrow \mathsf{MAC.Gen}(1^\lambda) \\ m \neq m_1 & (m, \mathsf{tag}) \leftarrow \mathcal{A}(1^\lambda)^{\mathsf{MAC.Tag}(\mathsf{s},\cdot)} \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ can access to the oracle only once and m_1 is the query from $\mathcal{A}$.

We have the following theorem.

Theorem 2.11. *There exists an information-theoretically secure OT-MAC.*

2.7 Non-interactive Zero-Knowledge Systems

We introduce the definition of a non-interactive zero-knowledge (NIZK) system and true-simulation extractability for it.

Definition 2.12 (NIZK). *Let L be an NP language associated with the corresponding NP relation R. A NIZK system for L is a tuple of algorithms* $(\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Vrfy})$.

- $\mathsf{NIZK.Setup}(1^\lambda)$*: The setup algorithm takes as input the security parameter* 1^λ *and outputs a common reference string* crs.
- $\mathsf{NIZK.Prove}(\mathsf{crs}, x, w)$*: The prove algorithm takes as input common reference string* crs*, NP instance* x*, and witness* w*, and outputs a proof* π.
- $\mathsf{NIZK.Vrfy}(\mathsf{crs}, x, \pi)$*: The verification algorithm takes as input common reference string* crs*, NP instance* x*, and proof* π*, and outputs* $\top$ *or* $\bot$.

Definition 2.13 (Completeness). *A NIZK system for NP is said to be complete if we have* $\mathsf{NIZK.Vrfy}(\mathsf{crs}, x, \mathsf{NIZK.Prove}(\mathsf{crs}, x, w)) = \top$ *for all common reference string* crs *output by* $\mathsf{NIZK.Setup}(1^\lambda)$ *and all valid statement/witness pairs* $(x, w) \in R$.

Definition 2.14 (True-Simulation Extractability). *Let* NIZK *be a NIZK system and* $\mathcal{A}$ *a QPT adversary. Let* $\mathsf{Sim} = (\mathsf{FkSetup}, \mathsf{Sim}_1, \mathsf{Sim}_2)$ *be a tuple of PPT algorithms. We define the following experiment* $\mathsf{Expt}^{\mathsf{se\text{-}real}}_{\mathcal{A},\mathsf{NIZK}}$.

1. *The challenger first generates* $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ *and sends* crs *to* $\mathcal{A}$.
2. $\mathcal{A}$ *sends* q *statement/witness pairs* $(x_i, w_i)_{i\in[q]}$ *to the challenger. The challenger responds with* $\{\pi_i\}_{i\in[q]}$*, where* $\pi_i \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, x_i, w_i)$ *for every* $i \in [q]$.
3. *Finally,* $\mathcal{A}$ *outputs* (x', π')*. The challenger outputs* 1 *if* $\mathsf{NIZK.Vrfy}(\mathsf{crs}, x', \pi') = \top$*,* $(x_i, w_i) \in R$ *for every* $i \in [q]$*, and* $x_i \neq x'$ *for every* $i \in [q]$ *hold. Otherwise, the challenger outputs* 0.

We also define the following experiment $\mathsf{Expt}^{\mathsf{se\text{-}sim}}_{\mathcal{A},\mathsf{Sim},\mathsf{NIZK}}$.

1. *The challenger first generates* $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{FkSetup}(1^\lambda)$ *and sends* crs *to* $\mathcal{A}$.
2. $\mathcal{A}$ *sends* q *statement/witness pairs* $(x_i, w_i)_{i\in[q]}$ *to the challenger. The challenger computes* $(\{\pi_i\}_{i\in[q]}, \mathsf{st}_{\mathsf{Sim}}) \leftarrow \mathsf{Sim}_1(\mathsf{crs}, \mathsf{td}, \{x_i\}_{i\in[q]})$ *and returns* $\{\pi_i\}_{i\in[q]}$ *to* $\mathcal{A}$.
3. *Finally,* $\mathcal{A}$ *outputs* (x', π')*. The challenger computes* $w' \leftarrow \mathsf{Sim}_2(\mathsf{st}_{\mathsf{Sim}}, x', \pi')$*. The challenger outputs* 1 *if* $\mathsf{NIZK.Vrfy}(\mathsf{crs}, x', \pi') = \top$*,* $(x_i, w_i) \in R$ *for every* $i \in [q]$*,* $(x', w') \in R$*, and* $x_i \neq x'$ *for every* $i \in [q]$ *hold. Otherwise, the challenger outputs* 0.

A NIZK system is said to be true-simulation extractable if for any QPT adversary $\mathcal{A}$*, there exists a tuple of PPT algorithms* Sim *such that we have*

$$\left|\Pr\left[1 \leftarrow \mathsf{Expt}^{\mathsf{se\text{-}real}}_{\mathcal{A},\mathsf{NIZK}}\right] - \Pr\left[1 \leftarrow \mathsf{Expt}^{\mathsf{se\text{-}sim}}_{\mathcal{A},\mathsf{Sim},\mathsf{NIZK}}\right]\right| \leq \mathsf{negl}(\lambda).$$

Ananth and La Placa [AL21] showed the following theorem.

Theorem 2.15. *There exists a true-simulation extractable NIZK system secure against polynomial (resp. sub-exponential) time quantum adversaries assuming the quantum hardness of the LWE problem against polynomial (resp. sub-exponential) time quantum adversaries.*

2.8 Noisy Trapdoor Claw-Free Hash Function

We recall the notion of noisy trapdoor claw-free (NTCF) hash function [BCM+18].

Definition 2.16 (NTCF Hash Function [BCM+18]). *Let $\mathcal{X}$, $\mathcal{Y}$ be finite sets, $\mathcal{D}_\mathcal{Y}$ the set of probability densities over $\mathcal{Y}$, and $\mathcal{K}_\mathcal{F}$ a finite set of keys. A family of functions*

$$\mathcal{F} := \{f_{\mathsf{k},b} : \mathcal{X} \to \mathcal{D}_\mathcal{Y}\}_{\mathsf{k}\in\mathcal{K}_\mathcal{F}, b\in\{0,1\}}$$

is a NTCF family if the following holds.

Efficient Function Generation: *There exists a PPT algorithm $\mathsf{NTCF.Gen}_\mathcal{F}$ which generates a key $\mathsf{k} \in \mathcal{K}_\mathcal{F}$ and a trapdoor td.*

Trapdoor Injective Pair: *For all keys $\mathsf{k} \in \mathcal{K}_\mathcal{F}$, the following holds.*

1. Trapdoor: *For all $b \in \{0,1\}$ and $x \neq x' \in \mathcal{X}$, $\mathsf{Supp}(f_{\mathsf{k},b}(x)) \cap \mathsf{Supp}(f_{\mathsf{k},b}(x')) = \emptyset$. In addition, there exists an efficient deterministic algorithm $\mathsf{Inv}_\mathcal{F}$ such that for all $b \in \{0,1\}, x \in \mathcal{X}$ and $y \in \mathsf{Supp}(f_{\mathsf{k},b}(x))$, $\mathsf{Inv}_\mathcal{F}(\mathsf{td}, b, y) = x$.*
2. Injective pair: *There exists a perfect matching relation $\mathcal{R}_\mathsf{k} \subseteq \mathcal{X} \times \mathcal{X}$ such that $f_{\mathsf{k},0}(x_0) = f_{\mathsf{k},1}(x_1)$ if and only if $(x_0, x_1) \in \mathcal{R}_\mathsf{k}$.*

Efficient Range Superposition: *For all keys $\mathsf{k} \in \mathcal{K}_\mathcal{F}$ and $b \in \{0,1\}$, there exists a function $f'_{\mathsf{k},b} : \mathcal{X} \to \mathcal{D}_\mathcal{Y}$ such that the following holds.*

1. *For all $(x_0, x_1) \in \mathcal{R}_\mathsf{k}$ and $y \in \mathsf{Supp}(f'_{\mathsf{k},b}(x_b))$, $\mathsf{Inv}_\mathcal{F}(\mathsf{td}, b, y) = x_b$ and $\mathsf{Inv}_\mathcal{F}(\mathsf{td}, b \oplus 1, y) = x_{b\oplus 1}$.*
2. *There exists an efficient deterministic procedure $\mathsf{Chk}_\mathcal{F}$ that takes as input $k, b \in \{0,1\}, x \in \mathcal{X}$ and $y \in \mathcal{Y}$ and outputs 1 if $y \in \mathsf{Supp}(f'_{\mathsf{k},b}(x))$ and 0 otherwise. This procedure does not need the trapdoor td.*
3. *For all $\mathsf{k} \in \mathcal{K}$ and $b \in \{0,1\}$,*

$$\mathop{\mathbb{E}}_{x\leftarrow\mathcal{X}}[\mathsf{H}^2(f_{\mathsf{k},b}(x), f'_{\mathsf{k},b}(x))] \leq \mathsf{negl}(\lambda).$$

Here H^2 is the Hellinger distance (See Section 2.2). In addition, there exists a QPT algorithm $\mathcal{S}amp_\mathcal{F}$ that takes as input k and $b \in \{0,1\}$ and prepare the quantum state

$$|\psi'\rangle = \frac{1}{\sqrt{|\mathcal{X}|}} \sum_{x\in\mathcal{X}, y\in\mathcal{Y}} \sqrt{(f'_{\mathsf{k},b}(x))(y)}\,|x\rangle\,|y\rangle\,.$$

This property and Lemma 2.1 immediately imply that

$$\left\| |\psi\rangle\langle\psi| - |\psi'\rangle\langle\psi'| \right\|_{\mathsf{tr}} \leq \mathsf{negl}(\lambda),$$

where $|\psi\rangle = \frac{1}{\sqrt{|\mathcal{X}|}} \sum_{x\in\mathcal{X}, y\in\mathcal{Y}} \sqrt{(f_{\mathsf{k},b}(x))(y)}\,|x\rangle\,|y\rangle$.

Adaptive Hardcore Bit: *For all keys $\mathsf{k} \in \mathcal{K}_\mathcal{F}$, the following holds. For some integer w that is a polynomially bounded function of λ,*

1. *For all $b \in \{0,1\}$ and $x \in \mathcal{X}$, there exists a set $G_{\mathsf{k},b,x} \subseteq \{0,1\}^w$ such that $\Pr_{d\leftarrow\{0,1\}^w}[d \notin G_{\mathsf{k},b,x}] \leq \mathsf{negl}(\lambda)$. In addition, there exists a PPT algorithm that checks for membership in $G_{\mathsf{k},b,x}$ given k, b, x, and td.*

2. *There is an efficiently computable injection $J : \mathcal{X} \to \{0,1\}^w$ such that J can be inverted efficiently on its range, and such that the following holds. Let*

$$H_{\mathsf{k}} := \{(b, x_b, d, d \cdot (J(x_0) \oplus J(x_1))) \mid b \in \{0,1\}, (x_0, x_1) \in \mathcal{R}_{\mathsf{k}}, d \in G_{\mathsf{k},0,x_0} \cap G_{\mathsf{k},1,x_1}\},$$
$$\overline{H}_{\mathsf{k}} := \{(b, x_b, d, c) \mid (b, x, d, c \oplus 1) \in H_k\},$$

then for any QPT $\mathcal{A}$, it holds that

$$\left| \Pr_{(\mathsf{k},\mathsf{td}) \leftarrow \mathsf{NTCF.Gen}_{\mathcal{F}}(1^\lambda)} [\mathcal{A}(\mathsf{k}) \in H_{\mathsf{k}}] - \Pr_{(\mathsf{k},\mathsf{td}) \leftarrow \mathsf{NTCF.Gen}_{\mathcal{F}}(1^\lambda)} [\mathcal{A}(\mathsf{k}) \in \overline{H}_{\mathsf{k}}] \right| \leq \mathsf{negl}(\lambda).$$

Brakerski et al. showed the following theorem.

Theorem 2.17 ([BCM+18]). *If we assume the quantum hardness of the LWE problem, then there exists an NTCF family.*

2.9 Secure Software Leasing

We introduce the notion of secure software leasing (SSL) defined by Ananth and La Placa [AL21].

Definition 2.18 (SSL with Setup [AL21]). *Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ be a circuit class such that $\mathcal{C}_\lambda$ contains circuits of input length n and output length m. A secure software lease scheme with setup for $\mathcal{C}$ is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Gen}, \mathcal{L}essor, \mathcal{R}un, \mathcal{C}heck)$.*

- $\mathsf{Setup}(1^\lambda)$*: The setup algorithm takes as input the security parameter 1^λ and outputs a classical string* crs.
- $\mathsf{Gen}(\mathsf{crs})$*: The key generation algorithm takes as input* crs *and outputs a secret key* sk.
- $\mathcal{L}essor(\mathsf{sk}, C)$*: The lease algorithm takes as input* sk *and a polynomial-sized classical circuit $C \in \mathcal{C}_\lambda$ and outputs a quantum state sft_C.*
- $\mathcal{R}un(\mathsf{crs}, \mathit{sft}_C, x)$*: The run algorithm takes as input* crs*, sft_C, and an input $x \in \{0,1\}^n$ for C, and outputs $y \in \{0,1\}^m$ and some state sft'. We use the notation $\mathcal{R}un_{\mathsf{out}}(\mathsf{crs}, \mathit{sft}_C, x) = y$ to denote that $\mathcal{R}un(\mathsf{crs}, \mathit{sft}_C, x)$ results in an output of the form (sft', y) for some state sft'.*
- $\mathcal{C}heck(\mathsf{sk}, \mathit{sft}^*_C)$*: The check algorithm takes as input* sk *and sft^*_C, and outputs $\top$ or $\bot$.*

Definition 2.19 (Correctness for SSL). *An SSL scheme $(\mathsf{Setup}, \mathsf{Gen}, \mathcal{L}essor, \mathcal{R}un, \mathcal{C}heck)$ for $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ is correct if for all $C \in \mathcal{C}_\lambda$, the following two properties hold:*

- *Correctness of $\mathcal{R}un$:*

$$\Pr\left[\forall x \ \Pr\left[\mathcal{R}un_{\mathsf{out}}(\mathsf{crs}, \mathit{sft}_C, x) = C(x)\right] \geq 1 - \mathsf{negl}(\lambda) \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sk} \leftarrow \mathsf{Gen}(\mathsf{crs}) \\ \mathit{sft}_C \leftarrow \mathcal{L}essor(\mathsf{sk}, C) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

- *Correctness of $\mathcal{C}heck$:*

$$\Pr\left[\mathcal{C}heck(\mathsf{sk}, \mathit{sft}_C) = \top \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sk} \leftarrow \mathsf{Gen}(\mathsf{crs}) \\ \mathit{sft}_C \leftarrow \mathcal{L}essor(\mathsf{sk}, C) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 2.20 (Reusability for SSL). *An SSL scheme* $(\mathsf{Setup}, \mathsf{Gen}, \mathcal{Lessor}, \mathcal{Run}, \mathcal{Check})$ *for* $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ *is reusable if for all* $C \in \mathcal{C}_\lambda$ *and for all* $x \in \{0,1\}^n$*, it holds that*

$$\left\| \mathit{sft}'_{C,x} - \mathit{sft}_C \right\|_{\mathrm{tr}} \leq \mathsf{negl}(\lambda),$$

where $\mathit{sft}'_{C,x}$ is the quantum state output by $\mathcal{Run}(\mathsf{crs}, \mathit{sft}_C, x)$.

Lemma 2.21 ([AL21]). *If an SSL scheme* $(\mathsf{Setup}, \mathsf{Gen}, \mathcal{Lessor}, \mathcal{Run}, \mathcal{Check})$ *for* $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ *is correct, then there exists a QPT algorithm* $\mathcal{Run}'$ *such that* $(\mathsf{Setup}, \mathsf{Gen}, \mathcal{Lessor}, \mathcal{Run}', \mathcal{Check})$ *is a reusable SSL scheme for* $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$.

Below, we introduce a security notion called finite-term lessor security for SSL. We can also consider a stronger security notion called infinite-term lessor security for SSL. For the definition of infinite-term lessor security, see the paper by Ananth and La Placa [AL21].

In the security experiment of SSL, an adversary outputs a bipartite state sft^* on the first and second registers. Let $\mathit{sft}^*_0 := \mathrm{Tr}_2[\mathit{sft}^*]$ and sft^*_0 is verified by $\mathcal{Check}$.[12] In addition, $P_2(\mathsf{sk}, \mathit{sft}^*)$ denotes the resulting post-measurement state on the second register (after the check on the first register). We write

$$P_2(\mathsf{sk}, \mathit{sft}^*) \propto \mathrm{Tr}_1[\Pi_1[(\mathcal{Check}(\mathsf{sk}, \mathit{sft}^*)_1 \otimes I_2)(\mathit{sft}^*)]]$$

for the state that $\mathcal{A}$ keeps after the first register has been returned and verified. Here, Π_1 denotes projecting the output of $\mathcal{Check}$ onto $\top$, and where $(\mathcal{Check}(\mathsf{sk}, \mathit{sft}^*)_1 \otimes I_2)(\mathit{sft}^*)$ denotes applying $\mathcal{Check}$ on to the first register, and the identity on the second register of sft^*.

Definition 2.22 (Perfect Finite-Term Lessor Security). *Let* β *be any inverse polynomial of* λ *and* $\mathcal{D}_\mathcal{C}$ *a distribution on* $\mathcal{C}$*. We define the* $(\beta, \mathcal{D}_\mathcal{C})$*-perfect finite-term lessor security game* $\mathsf{Expt}^{\mathsf{pft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_\mathcal{C}}(\lambda, \beta)$ *between the challenger and adversary* $\mathcal{A}$ *as follows.*

1. *The challenger generates* $C \leftarrow \mathcal{D}_\mathcal{C}$, $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{sk} \leftarrow \mathsf{Gen}(\mathsf{crs})$*, and* $\mathit{sft}_C \leftarrow \mathcal{Lessor}(\mathsf{sk}, C)$*, and sends* $(\mathsf{crs}, \mathit{sft}_C)$ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *outputs a bipartite state* sft^**. Below, we let* $\mathit{sft}^*_0 := \mathrm{Tr}_2[\mathit{sft}^*]$*.*
3. *If* $\mathcal{Check}(\mathsf{sk}, \mathit{sft}^*_0) = \top$ *and* $\forall x \ \Pr[\mathcal{Run}_{\mathsf{out}}(\mathsf{crs}, P_2(\mathsf{sk}, \mathit{sft}^*), x) = C(x)] \geq \beta$ *hold, where the probability is taken over the choice of the randomness of* $\mathcal{Run}$*, then the challenger outputs* 1*. Otherwise, the challenger outputs* 0*.*

We say that an SSL scheme $(\mathsf{Setup}, \mathsf{Gen}, \mathcal{Lessor}, \mathcal{Run}, \mathcal{Check})$ *is* $(\beta, \mathcal{D}_\mathcal{C})$*-perfect finite-term lessor secure, if for any QPT* $\mathcal{A}$ *that outputs a bipartite (possibly entangled) quantum state on the first and second registers, the following holds.*

$$\Pr\left[\mathsf{Expt}^{\mathsf{pft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_\mathcal{C}}(\lambda, \beta) = 1\right] \leq \mathsf{negl}(\lambda).$$

In addition to the above perfect finite-term lessor security, we also introduce a new security notion *average-case finite-term lessor security*. For an SSL scheme for a family of PRF, we consider average-case finite-term lessor security. This is because when we consider cryptographic functionalities, the winning condition "$\forall x \ \Pr[\mathcal{Run}_{\mathsf{out}}(\mathsf{crs}, P_2(\mathsf{sk}, \sigma^*), x) = C(x)] \geq \beta$" posed to the adversary in the definition of perfect finite-term lessor security seems to be too strong. In fact, for those functionalities, adversaries who can generate a bipartite state sft^* such that $\mathcal{Run}_{\mathsf{out}}(\mathsf{crs}, P_2(\mathsf{sk}, \mathit{sft}^*), x) = C(x)$ holds for some fraction of inputs x should be regarded as successful adversaries. Average-case finite-term lessor security considers those adversaries.

[12] $\mathrm{Tr}_i[\mathcal{X}]$ is the partial trace of $\mathcal{X}$ where the i-th register is traced out.

Definition 2.23 (Average-Case Finite-Term Lessor Security). *Let ϵ be any inverse polynomial of λ and $\mathcal{D}_{\mathcal{C}}$ a distribution on $\mathcal{C}$. We define the $(\epsilon, \mathcal{D}_{\mathcal{C}})$-average-case finite-term lessor security game $\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda,\epsilon)$ between the challenger and adversary by replacing the third stage of $\mathsf{Expt}^{\mathsf{pft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda,\beta)$ with the following.*

3. *If $\mathcal{C}heck(\mathsf{sk}, \mathit{sft}^*_0) = \top$ and $\Pr[\mathcal{R}un_{\mathsf{out}}(\mathsf{crs}, P_2(\mathsf{sk}, \mathit{sft}^*), x) = C(x)] \geq \epsilon$ hold, where the probability is taken over the choice of $x \leftarrow \{0,1\}^n$ and the random coin of $\mathcal{R}un$, then the challenger outputs 1. Otherwise, the challenger outputs 0.*

We say that an SSL scheme $(\mathsf{Setup}, \mathsf{Gen}, \mathcal{L}essor, \mathcal{R}un, \mathcal{C}heck)$ is $(\epsilon, \mathcal{D}_{\mathcal{C}})$-average-case finite-term lessor secure, if for any QPT $\mathcal{A}$ that outputs a bipartite (possibly entangled) quantum state on the first and second registers, the following holds.

$$\Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda,\epsilon) = 1\right] \leq \mathsf{negl}(\lambda).$$

3 Two-Tier Quantum Lightning

In this section, we present definitions of our new tools and their instantiations.

3.1 Two-Tier Quantum Lightning

We define two-tier QL, which is a weaker variant of QL [Zha21]. A big difference from QL is that we have two types of verification called semi-verification and full-verification. We need a secret key for full-verification while we use a public key for semi-verification.

Definition 3.1 (Two-Tier Quantum Lightning (syntax)). *A two-tier quantum lightning scheme is a tuple of algorithms $(\mathsf{Setup}, \mathcal{B}oltGen, \mathcal{S}emiVrfy, \mathcal{F}ullVrfy)$.*

- $\mathsf{Setup}(1^\lambda)$*: The setup algorithm takes as input the security parameter 1^λ and outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.*
- $\mathcal{B}oltGen(\mathsf{pk})$*: The bolt generation algorithm takes as input* pk *and outputs a classical string* snum *(called a serial number) and a quantum state bolt (called a bolt for the serial number).*
- $\mathcal{S}emiVrfy(\mathsf{pk}, \mathsf{snum}, \mathit{bolt})$*: The semi-verification algorithm takes as input* pk, snum, *and bolt and outputs $(\top, \mathit{bolt}')$ or $\bot$.*
- $\mathcal{F}ullVrfy(\mathsf{sk}, \mathsf{snum}, \mathit{bolt})$*: The full-verification algorithm takes as input* sk, snum, *and bolt and outputs $\top$ or $\bot$.*

Definition 3.2 (Correctness for Two-Tier Quantum Lightning). *There are two verification processes. We say that a two-tier quantum lightning with classical verification is correct if it satisfies the following two properties.*

Semi-verification correctness:

$$\Pr\left[(\top, \mathit{bolt}') \leftarrow \mathcal{S}emiVrfy(\mathsf{pk}, \mathsf{snum}, \mathit{bolt}) \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)\\ (\mathsf{snum}, \mathit{bolt}) \leftarrow \mathcal{B}oltGen(\mathsf{pk})\end{array}\right] > 1 - \mathsf{negl}(\lambda).$$

Full-verification correctness:

$$\Pr\left[\top \leftarrow \mathcal{F}ullVrfy(\mathsf{sk}, \mathsf{snum}, \mathit{bolt}) \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)\\ (\mathsf{snum}, \mathit{bolt}) \leftarrow \mathcal{B}oltGen(\mathsf{pk})\end{array}\right] > 1 - \mathsf{negl}(\lambda).$$

Definition 3.3 (Reusability for Two-Tier Quantum Lightning). *A two-tier quantum lightning scheme* $(\mathsf{Setup}, \mathcal{BoltGen}, \mathcal{SemiVrfy}, \mathcal{FullVrfy})$ *is reusable if for all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{snum}, \mathit{bolt}) \leftarrow \mathcal{BoltGen}(\mathsf{pk})$, *and* $(\mathit{bolt}', \top) \leftarrow \mathcal{SemiVrfy}(\mathsf{pk}, \mathsf{snum}, \mathit{bolt})$, *it holds that*

$$\left\| \mathit{bolt}' - \mathit{bolt} \right\|_{\mathrm{tr}} \leq \mathsf{negl}(\lambda).$$

Remark 3.4. We can show that any two-tier QL scheme that satisfies semi-verification correctness can be transformed into one that satisfies reusability by using the Almost As Good As New Lemma [Aar05] similarly to an analogous statement for SSL shown in [AL21]. Therefore, we focus on correctness.

Definition 3.5 (Two-Tier Unclonability). *We define the two-tier unclonability game between a challenger and an adversary* $\mathcal{A}$ *as follows.*

1. *The challenger generate* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* pk *to* $\mathcal{A}$.
2. $\mathcal{A}$ *outputs possibly entangled quantum states* $\mathcal{L}_0$ *and* $\mathcal{L}_1$ *and a classical string* snum^*, *and sends them to the challenger.*
3. *The challenger runs* $\mathcal{FullVrfy}(\mathsf{sk}, \mathsf{snum}^*, \mathcal{L}_0)$ *and* $\mathcal{SemiVrfy}(\mathsf{pk}, \mathsf{snum}^*, \mathcal{L}_1)$. *If both the outputs are* $\top$, *then this experiments outputs* 1. *Otherwise, it outputs* 0.

This game is denoted by $\mathsf{Exp}^{\mathsf{tt\text{-}unclone}}_{\mathcal{A},\Sigma}(1^\lambda)$. *A two-tier quantum lightning scheme is two-tier unclonable if for any QPT adversary* $\mathcal{A}$, *it holds that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{tt\text{-}unclone}}_{\mathcal{A},\Sigma}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

Definition 3.6 (Secure Two-Tier Quantum Lightning). *A two-tier quantum lightning scheme is secure if it satisfies Definitions 3.1 to 3.3 and 3.5.*

3.2 Two-Tier Quantum Lightning from SIS

We show how to construct a two-tier quantum lightning scheme from the SIS assumption. The construction is based on the franchised quantum money scheme by Roberts and Zhandry [RZ21]. They (implicitly) proved the following lemma holds by appropriately setting parameters n, m, q, β in such a way that $\mathrm{SIS}_{n,m,q,\beta}$ is believed to be hard:

Lemma 3.7 ([RZ21]). *There exist PPT algorithm* $\mathsf{TrapGen}$ *and QPT algorithms* $(\mathcal{FQMGen}, \mathcal{FQMVrfy})$ *that work as follows:*[13]

$\mathsf{TrapGen}(1^\lambda)$**:** *This algorithm generates a matrix* $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ *and its trapdoor* td.

$\mathcal{FQMGen}(\boldsymbol{A})$**:** *Given a matrix* $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$, *it outputs a vector* $\boldsymbol{y} \in \mathbb{Z}_q^n$ *along with a quantum state*

$$|\Sigma\rangle = \sum_{\boldsymbol{x} \in \mathbb{Z}_q^m : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \bmod q} \sqrt{p(\boldsymbol{x})}\, |\boldsymbol{x}\rangle\,.$$

for a certain probability density function p *over* $\{\boldsymbol{x} \in \mathbb{Z}_q^m : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \bmod q\}$ *such that if we take* $\boldsymbol{x}$ *according to* p, *we have* $\Pr[\|\boldsymbol{x}\| > \beta/2] = \mathsf{negl}(\lambda)$. [14]

$\mathcal{FQMVrfy}(\mathsf{td}, \boldsymbol{y}, |\Sigma\rangle)$**:** *It outputs* $\top$ *or* $\bot$.

Moreover, the following is satisfied:

[13] $\mathsf{TrapGen}$ is by now a standard algorithm to sample a matrix with its trapdoor [GPV08, MP12].

[14] Specifically, p is proportional to discrete Gaussian.

1. *If we generate* $(\boldsymbol{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda)$ *and* $(\boldsymbol{y}, |\Sigma\rangle) \leftarrow \mathcal{FQMGen}(\boldsymbol{A})$*, we have*

$$\Pr[\mathcal{FQMVrfy}(\mathsf{td}, \boldsymbol{y}, |\Sigma\rangle) = \bot] = \mathsf{negl}(\lambda).$$

2. *For any* $(\boldsymbol{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda)$*,* $\boldsymbol{y} \in \mathbb{Z}_q^n$ *and (possibly malformed) quantum state sigma, suppose that we have* $\mathcal{FQMVrfy}(\mathsf{td}, \boldsymbol{y}, \mathit{sigma}) = \top$*, and let sigma′ be the state after running* $\mathcal{FQMVrfy}$*. If we measure sigma′, the outcome* $\boldsymbol{x} \in \mathbb{Z}_q^m$ *satisfies* $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \bmod q$*, and no value for* $\boldsymbol{x} \in \mathbb{Z}_q^m$ *has overwhelming probability of being measured.*

Construction 3.8. Our two-tier quantum lightning scheme is described as follows.

- $\mathsf{Setup}(1^\lambda)$: Run $(\boldsymbol{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda)$ and outputs $\mathsf{pk} := \boldsymbol{A}$ and $\mathsf{sk} := \mathsf{td}$.
- $\mathcal{BoltGen}(\mathsf{pk} = \boldsymbol{A})$: Run $(\boldsymbol{y}, |\Sigma\rangle) \leftarrow \mathcal{FQMGen}(\boldsymbol{A})$ and outputs $(\mathsf{snum}, \mathit{bolt}) := (\boldsymbol{y}, |\Sigma\rangle)$.
- $\mathcal{FullVrfy}(\mathsf{sk} = \mathsf{td}, \mathsf{snum} = \boldsymbol{y}, \mathit{bolt})$: This is exactly the same algorithm as $\mathcal{FQMVrfy}(\mathsf{td}, \boldsymbol{y}, \mathit{bolt})$.
- $\mathcal{SemiVrfy}(\mathsf{pk} = \boldsymbol{A}, \mathsf{snum} = \boldsymbol{y}, \mathit{bolt})$: This algorithm checks if the value $\boldsymbol{x}$ in the register of *bolt* satisfies $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ and $\|\boldsymbol{x}\| \leq \beta/2$ in superposition by writing the result into another register and measuring it. If that is satisfied, then it outputs $\top$ along with a resulting state bolt' in the register that stored *bolt*. Otherwise, it outputs $\bot$.

Full- and semi-verification correctness directly follows from Lemma 3.7. Security is stated as follows:

Theorem 3.9. *If we assume the quantum hardness of the* $\mathrm{SIS}_{n,m,q,\beta}$*, then the above two-tier quantum lightning satisfies two-tier unclonability.*

Proof. We show that if two-tier unclonability of Construction 3.8 is broken, then the SIS problem is also broken. We construct a QPT adversary $\mathcal{B}$ for SIS by using a QPT adversary $\mathcal{A}$ against two-tier QL. $\mathcal{B}$ is given a matrix $\boldsymbol{A}$ and sends $\mathsf{pk} := \boldsymbol{A}$ to $\mathcal{A}$. When $\mathcal{A}$ outputs $(\mathsf{snum}^*, \mathcal{L}_0, \mathcal{L}_1)$, $\mathcal{B}$ runs $(b_0, \mathcal{L}_0') \leftarrow \mathcal{SemiVrfy}(\mathsf{pk}, \mathsf{snum}^*, \mathcal{L}_0)$ and $b_1 \leftarrow \mathcal{FullVrfy}(\mathsf{pk}, \mathsf{snum}^*, \mathcal{L}_1)$ and let $\mathcal{L}_1'$ be the resulting state in the register that stored $\mathcal{L}_1$. If $b_0 = \bot$ or $b_1 = \bot$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ measures $\mathcal{L}_0'$ and $\mathcal{L}_1'$. Let the results of the measurement $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, respectively. Then $\mathcal{B}$ outputs $\boldsymbol{x}_0 - \boldsymbol{x}_1$.

By definitions of $\mathcal{FullVrfy}$ and $\mathcal{SemiVrfy}$ and Lemma 3.7, we have $\boldsymbol{A}\boldsymbol{x}_b = \mathsf{snum}^*$ and $\|\boldsymbol{x}_b\| \leq \beta/2$ for both $b \in \{0,1\}$. Therefore, $\mathcal{B}$ succeeds in solving the SIS problem as long as $\boldsymbol{x}_0 \neq \boldsymbol{x}_1$. Again, Lemma 3.7 ensures that we have $\boldsymbol{x}_0 \neq \boldsymbol{x}_1$ with non-negligible probability. This completes the proof. ■

3.3 Two-Tier Quantum Lightning with Classical Verification

We extend two-tier QL to have an algorithm that converts a bolt into a classical certificate which certifies that the bolt was collapsed. This bolt-to-certificate capability was introduced by Coladangelo and Sattath [CS20] for the original QL notion. We can consider a similar notion for two-tier QL.

Definition 3.10 (Two-tier Quantum Lightning with Classical Verification (syntax)). *A two-tier quantum lightning scheme with classical semi-verification is a tuple of algorithms* $(\mathsf{Setup}, \mathcal{BoltGen}, \mathcal{BoltCert}, \mathcal{SemiVrfy}, \mathsf{CertVrfy})$.

- $\mathsf{Setup}(1^\lambda)$*: The setup algorithm takes as input the security parameter* 1^λ *and outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$.
- $\mathcal{BoltGen}(\mathsf{pk})$*: The bolt generation algorithm takes as input* pk *and outputs a classical string* snum *(called a serial number) and a quantum state bolt (called a bolt for the serial number).*
- $\mathcal{SemiVrfy}(\mathsf{pk}, \mathsf{snum}, \mathit{bolt})$*: The semi-verification algorithm takes as input* pk*,* snum*, and bolt and outputs* $(\top, \mathit{bolt}')$ *or* $\bot$.

- $\mathcal{BoltCert}(\mathit{bolt})$: *The bolt certification algorithm takes as input* bolt *and outputs a classical string* cert *(called a certification for collapsing a bolt).*
- $\mathsf{CertVrfy}(\mathsf{sk}, \mathsf{snum}, \mathsf{cert})$: *The certification-verification algorithm takes as input* sk *and* cert *and outputs* $\top$ *or* $\bot$.

Definition 3.11 (Correctness for Two-Tier Quantum Lighting with Classical Verification). *There are two verification processes. We say that a two-tier quantum lightning with classical verification is correct if it satisfies the following two properties.*

Semi-verification correctness: *It holds that*

$$\Pr\left[\begin{array}{l|l} (\top, \mathit{bolt}') \leftarrow \mathcal{SemiVrfy}(\mathsf{pk}, \mathsf{snum}, \mathit{bolt}) & \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{snum}, \mathit{bolt}) \leftarrow \mathcal{BoltGen}(\mathsf{pk})\end{array}\end{array}\right] > 1 - \mathsf{negl}(\lambda).$$

Certification-verification correctness: *It holds that*

$$\Pr\left[\begin{array}{l|l} \top \leftarrow \mathsf{CertVrfy}(\mathsf{sk}, \mathsf{snum}, \mathsf{cert}) & \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{snum}, \mathit{bolt}) \leftarrow \mathcal{BoltGen}(\mathsf{pk}) \\ \mathsf{cert} \leftarrow \mathcal{BoltCert}(\mathit{bolt})\end{array}\end{array}\right] > 1 - \mathsf{negl}(\lambda).$$

Definition 3.12 (Reusability for Two-Tier Quantum Lighting with Classical Verification). *A two-tier quantum lightning scheme with classical verification* $(\mathsf{Setup}, \mathcal{BoltGen}, \mathcal{SemiVrfy}, \mathcal{BoltCert}, \mathsf{CertVrfy})$ *is reusable if for all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{snum}, \mathit{bolt}) \leftarrow \mathcal{BoltGen}(\mathsf{pk})$, *and* $(\mathit{bolt}', \top) \leftarrow \mathcal{SemiVrfy}(\mathsf{pk}, \mathsf{snum}, \mathit{bolt})$, *it holds that*

$$\left\| \mathit{bolt} - \mathit{bolt}' \right\|_{\mathrm{tr}} \leq \mathsf{negl}(\lambda).$$

Remark 3.13. Similarly to Remark 3.4, any two-tier QL scheme with classical verification that satisfies semi-verification correctness can be transformed into one that satisfies reusability. Therefore, we focus on correctness.

Definition 3.14 (Two-Tier Unclonability with Classical Verification). *We define the two-tier unclonability game between a challenger and an adversary* $\mathcal{A}$ *in the classical verification setting as follows.*

1. *The challenger generates* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(\mathsf{snum}, \mathit{bolt}) \leftarrow \mathcal{BoltGen}(\mathsf{pk})$ *and sends* pk *to* $\mathcal{A}$.
2. $\mathcal{A}$ *outputs a classical string* snum, *a quantum state* $\mathcal{L}$, *and a classical string* CL *and sends them to the challenger.*
3. *The challenger runs* $\mathsf{CertVrfy}(\mathsf{sk}, \mathsf{snum}, \mathsf{CL})$ *and* $\mathcal{SemiVrfy}(\mathsf{pk}, \mathsf{snum}, \mathcal{L})$. *If both the outputs are* $\top$, *then this experiments outputs* 1. *Otherwise, it outputs* 0.

This game is denoted by $\mathsf{Exp}^{\mathsf{tt\text{-}unclone\text{-}cv}}_{\mathcal{A},\Sigma}(1^\lambda)$.

We say that $\Sigma = (\mathsf{Setup}, \mathcal{BoltGen}, \mathcal{SemiVrfy}, \mathcal{BoltCert}, \mathsf{CertVrfy})$ *is two-tier unclonable if the following holds. For any QPT adversary* $\mathcal{A}$, *it holds that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{tt\text{-}unclone\text{-}cv}}_{\mathcal{A},\Sigma}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

Definition 3.15 (Secure Two-Tier Quantum Lightning with Classical Verification). *A two-tier quantum lightning with classical verification is secure if it satisfies Definitions 3.10 to 3.12 and 3.14.*

Note that a two-tier quantum lightning scheme with classical verification can be easily transformed into an ordinary two-tier quantum lightning scheme. This is done by setting the latter's full-verification algorithm as the combination of the bolt certification algorithm and the certification-verification algorithm of the former. Namely, we have the following theorem.

Theorem 3.16. *If there exists two-tier quantum lightning with classical verification, then there also exists ordinary two-tier quantum lightning.*

3.4 Two-Tier Quantum Lightning with Classical Verification from LWE

In this section, we show how to construct a two-tier QL scheme with classical verification from the LWE assumption. First, we define an amplified version of the adaptive hardcore bit property of an NTCF family.

Definition 3.17 (Amplified Adaptive Hardcore Property). *We say that a NTCF family $\mathcal{F}$ (defined in Definition 2.16) satisfies the amplified adaptive hardcore property if for any QPT $\mathcal{A}$ and $n = \omega(\log\lambda)$, it holds that*

$$\Pr\left[\begin{array}{l|l} \forall i\in[n]\ x_i = x_{i,b_i}, & (\mathsf{k}_i,\mathsf{td}_i)\leftarrow \mathsf{NTCF.Gen}_{\mathcal{F}}(1^\lambda)\ \textit{for}\ i\in[n] \\ d_i\in G_{\mathsf{k},0,x_{i,0}}\cap G_{\mathsf{k},1,x_{i,1}}, & (\{(b_i,x_i,y_i,d_i,m_i)\}_{i\in[n]})\leftarrow \mathcal{A}(\{\mathsf{k}_i\}_{i\in[n]}) \\ m_i = d_i\cdot(J(x_{i,0})\oplus J(x_{i,1})) & x_{i,\beta}\leftarrow \mathsf{Inv}_{\mathcal{F}}(\mathsf{td}_i,\beta,y_i)\ \textit{for}\ (i,\beta)\in[n]\times\{0,1\} \end{array}\right] = \mathsf{negl}(\lambda).$$

As implicitly shown in [RS19], any NTCF family satisfies the amplified adaptive hardcore property.[15]

Lemma 3.18 (Implicit in [RS19]). *Any NTCF family satisfies the amplified adaptive hardcore property.*

Proof. (sketch.) This proof sketch is a summary of the proof in [RS19]. Canetti et al. [CHS05] proved that a parallel repetition exponentially decreases hardness of *weakly verifiable puzzle*, which is roughly a computational problem whose solution can be verified by a secret verification key generated along with the problem. Though Canetti et al. only considered hardness against classical algorithms, Radian and Sattath [RS19] observed that a similar result holds even for quantum algorithms. Then we consider a weakly verifiable puzzle described below:

1. A puzzle generation algorithm runs $(\mathsf{k},\mathsf{td})\leftarrow\mathsf{NTCF.Gen}_{\mathcal{F}}(1^\lambda)$ and publishes k as a puzzle while keeping td as a secret verification key.
2. We say that (b,x,y,d,m) is a valid solution to the puzzle k if it holds that $x = x_b$, $d\in G_{\mathsf{k},0,x_0}\cap G_{\mathsf{k},1,x_1}$, and $m = d\cdot(J(x_0)\oplus J(x_1))$ where $x_\beta\leftarrow\mathsf{Inv}_{\mathcal{F}}(\mathsf{td},\beta,y)$ for $\beta\in\{0,1\}$.

We can see that the adaptive hardcore property implies that a QPT algorithm can find a valid solution of the above weakly verifiable puzzle with probability at most $\frac{1}{2}+\mathsf{negl}(\lambda)$. By applying the amplification theorem of [CHS05, RS19] as explained above, $n=\omega(\log(\lambda))$-parallel repetition version of the above protocol is hard for any QPT algorithm to solve with non-negligible probability. This is just a rephrasing of amplified adaptive hardcore property. ■

Two-Tier Quantum Lightning from NTCF. We show how to construct a two-tier QL scheme with classical verification from an NTCF family.

Construction 3.19. Let $n=\omega(\log\lambda)$. Our two-tier QL with classical verification scheme is described as follows.

- $\mathsf{Setup}(1^\lambda)$: Generate $(\mathsf{k}_i,\mathsf{td}_i)\leftarrow\mathsf{NTCF.Gen}_{\mathcal{F}}(1^\lambda)$ for $i\in[n]$ and set $(\mathsf{pk},\mathsf{sk}):=(\{\mathsf{k}_i\}_{i\in[n]},\{\mathsf{td}_i\}_{i\in[n]})$.
- $\mathcal{BoltGen}(\mathsf{pk})$: Parse $\mathsf{pk}=\{\mathsf{k}_i\}_{i\in[n]}$. For each $i\in[n]$, generate a quantum state

$$|\psi_i'\rangle = \frac{1}{\sqrt{|\mathcal{X}|}}\sum_{x\in\mathcal{X},y\in\mathcal{Y},b\in\{0,1\}}\sqrt{(f'_{\mathsf{k}_i,b}(x))(y)}\,|b,x\rangle\,|y\rangle$$

  by using $\mathcal{Samp}_{\mathcal{F}}$, measure the last register to obtain $y_i\in\mathcal{Y}$, and let $|\phi_i'\rangle$ be the post-measurement state where the measured register is discarded. Output $(\mathsf{snum},\mathcal{bolt}):=(\{y_i\}_{i\in[n]},\{|\phi'_i\rangle\}_{i\in[n]})$.

[15] [RS19] proved essentially the same lemma through an abstraction which they call *1-of-2 puzzle*.

- $\mathit{SemiVrfy}(\mathsf{pk}, \mathsf{snum}, \mathit{bolt})$: Parse $\mathsf{pk} = \{\mathsf{k}_i\}_{i\in[n]}$, $\mathsf{snum} = \{y_i\}_{i\in[n]}$, $\mathit{bolt} = \{\mathit{bolt}_i\}_{i\in[n]}$. For each $i \in [n]$, check if the value (b_i, x_i) in the register of bolt_i satisfies $y \in \mathsf{Supp}(f'_{\mathsf{k}_i,b_i}(x_i))$ in superposition by writing the result to another register and measuring it. We note that this procedure can be done efficiently without using td_i since $y \in \mathsf{Supp}(f'_{\mathsf{k}_i,b_i}(x_i))$ can be publicly checked by using $\mathsf{Chk}_{\mathcal{F}}$ as defined in Definition 2.16. If the above verification passes for all $i \in [n]$, then output $\top$ and the post-measurement state (discarding measured registers). Otherwise, output $\bot$.
- $\mathit{BoltCert}(\mathit{bolt})$: Parse $\mathit{bolt} = \{\mathit{bolt}_i\}_{i\in[n]}$. For each $i \in [n]$, do the following: Evaluate the function J on the second register of bolt_i. That is, apply a unitary that maps $|b, x\rangle$ to $|b, J(x)\rangle$ to bolt_i. (Note that this can be done efficiently since J is injective and efficiently invertible.) Then, apply Hadamard transform and measure both registers to obtain (m_i, d_i). Output $\mathsf{cert} := \{(d_i, m_i)\}_{i\in[n]}$.
- $\mathsf{CertVrfy}(\mathsf{sk}, \mathsf{snum}, \mathsf{cert})$: Parse $\mathsf{sk} = \{\mathsf{td}_i\}_{i\in[n]}$, $\mathsf{snum} = \{y_i\}_{i\in[n]}$, and $\mathsf{cert} = \{(d_i, m_i)\}_{i\in[n]}$. For each $i \in [n]$ and $\beta \in \{0,1\}$, compute $x_{i,\beta} \leftarrow \mathsf{Inv}_{\mathcal{F}}(\mathsf{td}_i, \beta, y_i)$. Output $\top$ if and only if it holds that $d_i \in G_{\mathsf{k},0,x_{i,0}} \cap G_{\mathsf{k},1,x_{i,1}}$ and $m_i = d_i \cdot (J(x_{i,0}) \oplus J(x_{i,1}))$ for all $i \in [n]$.

Theorem 3.20. *If there exists an NTCF family, there exists a two-tier QL with classical verification.*

Proof of Theorem 3.20. We prove correctness and two-tier unclonability below:

Correctness of certification-verification. We need to prove that if cert is generated by $\mathit{BoltCert}(\mathit{bolt})$ for an honestly generated bolt corresponding a serial number snum, $\mathsf{CertVrfy}(\mathsf{sk}, \mathsf{snum}, \mathsf{cert})$ returns $\top$ with overwhelming probability.

For each $i \in [n]$, if we define a quantum state

$$|\psi_i\rangle = \frac{1}{\sqrt{|\mathcal{X}|}} \sum_{x\in\mathcal{X}, y\in\mathcal{Y}, b\in\{0,1\}} \sqrt{(f_{\mathsf{k}_i,b}(x))(y)}\, |b, x\rangle\, |y\rangle\,,$$

then we have

$$\left\| |\psi_i\rangle\langle\psi_i| - |\psi'_i\rangle\langle\psi'_i| \right\|_{\mathrm{tr}} \leq \mathsf{negl}(\lambda),$$

as observed in Definition 2.16 (where we used Lemma 2.1). Therefore, even if we replace $|\psi'_i\rangle$ with $|\psi_i\rangle$ for each $i \in [n]$ in the execution of $\mathit{BoltGen}(\mathsf{pk})$ to generate bolt, the probability that $\mathsf{CertVrfy}(\mathsf{sk}, \mathsf{snum}, \mathsf{cert})$ returns $\top$ only negligibly changes. Therefore, it suffices to prove that $\mathsf{CertVrfy}(\mathsf{sk}, \mathsf{snum}, \mathsf{cert})$ returns $\top$ with overwhelming probability in a modified experiment where $|\psi'_i\rangle$ is replaced with $|\psi_i\rangle$ for each $i \in [n]$.[16] In this experiment, if we let bolt_i be the i-th component of bolt, then we have

$$\mathit{bolt}_i = \frac{1}{\sqrt{2}}(|0, x_{i,0}\rangle + |1, x_{i,1}\rangle)$$

for each $i \in [n]$ where $x_{i,\beta} \leftarrow \mathsf{Inv}_{\mathcal{F}}(\mathsf{td}_i, \beta, y_i)$ for $\beta \in \{0,1\}$ by the injective property of $\mathcal{F}$. If we apply J to the second register of bolt_i and then apply Hadamard transform for both registers as in $\mathit{BoltCert}$, then the resulting state can be written as

$$\begin{aligned}
&2^{-\frac{w+2}{2}} \sum_{d,b,m} (-1)^{d\cdot J(x_{i,b})\oplus mb} |m\rangle\, |d\rangle \\
&= 2^{-\frac{w}{2}} \sum_{d\in\{0,1\}^w} (-1)^{d\cdot J(x_{i,0})} |d \cdot (J(x_{i,0}) \oplus J(x_{i,1}))\rangle\, |d\rangle\,.
\end{aligned}$$

Therefore, the measurement result is (m_i, d_i) such that $m_i = d_i \cdot (J(x_{i,0}) \oplus J(x_{i,1}))$ for a uniform $d_i \leftarrow \{0,1\}^w$. By the adaptive hardcore bit property (the first item) in Definition 2.16, it holds that $d_i \in G_{\mathsf{k}_i,0,x_{i,0}} \cap G_{\mathsf{k}_i,1,x_{i,1}}$ except negligible probability. Therefore, the certificate $\mathsf{cert} = \{(d_i, m_i)\}_{i\in[n]}$ passes the verification by $\mathsf{CertVrfy}$ with overwhelming probability.

[16] Of course, such a replacement cannot be done efficiently. We consider such an experiment only as a proof tool.

Correctness of semi-verification. Let $\mathit{bolt} = \{\phi'_i\}_{i\in[n]}$ be an honestly generated bolt. By the definition of $\mathcal{BoltGen}$, $|\phi_i\rangle$ is a superposition of (b,x) such that $y \in \mathsf{Supp}(f'_{\mathsf{k}_i,b}(x))$. This clearly passes the verification by $\mathcal{SemiVrfy}$.

Two-tier unclonability. As shown in Lemma 3.18, any NTCF family satisfies the amplified adaptive hardcore property. We show that if there exists a QPT adversary $\mathcal{A}$ that breaks the two-tier unclonability with classical verification of Construction 3.19 with probability ϵ, we can construct a QPT adversary $\mathcal{B}$ that breaks the amplified adaptive hardcore property the NTCF with probability ϵ.

$\mathcal{B}$ is given $\{\mathsf{k}_i\}_{i\in[n]}$ and sends $\mathsf{pk} := \{\mathsf{k}_i\}_{i\in[n]}$ to $\mathcal{A}$ this implicitly sets $\mathsf{sk} := \{\mathsf{td}_i\}_{i\in[n]}$). When $\mathcal{A}$ outputs $(\mathsf{snum}, \mathcal{L}, \mathsf{cert})$, $\mathcal{B}$ parses $\mathsf{snum} = \{y_i\}_{i\in[n]}$, $\mathcal{L} = \{\mathcal{L}_i\}_{i\in[n]}$, and $\mathsf{cert} = \{(d_i, m_i)\}_{i\in[n]}$, measures $\mathcal{L}_i$ to obtain (b_i, x_i) for each $i \in [n]$, and outputs $\{(b_i, x_i, y_i, d_i, m_i)\}_{i\in[n]}$.

By assumption on $\mathcal{A}$, it holds that $\mathcal{SemiVrfy}(\mathsf{pk}, \mathsf{snum}, \mathcal{L}) = \top$ and $\mathsf{CertVrfy}(\mathsf{sk}, \mathsf{snum}, \mathsf{cert}) = \top$ with probability ϵ. If $\mathcal{SemiVrfy}(\mathsf{pk}, \mathsf{snum}, \mathcal{L}) = \top$ holds, we have $y_i \in \mathsf{Supp}(f'_{\mathsf{k}_i,b_i}(x_i))$ for each $i \in [n]$ by the construction of $\mathcal{SemiVrfy}$. We note that $y_i \in \mathsf{Supp}(f'_{\mathsf{k}_i,b_i}(x_i))$ implies $x_i = x_{i,b_i}$ by the efficient range superposition property of Definition 2.16 where $x_{i,\beta} \leftarrow \mathsf{Inv}_{\mathcal{F}}(\mathsf{td}_i, \beta, y_i)$ for $\beta \in \{0,1\}$. If $\mathsf{CertVrfy}(\mathsf{sk}, \mathsf{snum}, \mathsf{cert}) = \top$ we have $d_i \in G_{\mathsf{k},0,x_{i,0}} \cap G_{\mathsf{k},1,x_{i,1}}$ and $m_i = d_i \cdot (J(x_{i,0}) \oplus J(x_{i,1}))$ for all $i \in [n]$. Clearly, $\mathcal{B}$ wins the amplified adaptive hardcore game when both of them happen, which happens with probability ϵ by the assumption. This completes the proof. ■

By combining Theorems 2.17 and 3.20, the following corollary immediately follows.

Corollary 3.21. *If we assume the quantum hardness of the LWE problem, there exists a secure two-tier QL with classical verification.*

4 Relaxed Watermarking

In this section, we introduce the notion of relaxed watermarking and concrete constructions of relaxed watermarking.

4.1 Definition of Relaxed Watermarking

We introduce the definition of relaxed watermarking. The following definition captures publicly markable and extractable watermarking schemes. After the definition, we state the difference between relaxed watermarking and classical cryptographic watermarking [CHN+18].

Definition 4.1 (Relaxed Watermarking Syntax). *Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ be a circuit class such that $\mathcal{C}_\lambda$ contains circuits of input length is n and output length m. A relaxed watermarking scheme for the circuit class $\mathcal{C}$ and a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ consists of four PPT algorithms* $(\mathsf{Gen}, \mathsf{Mark}, \mathsf{Extract}, \mathsf{Eval})$.

Key Generation: $\mathsf{Gen}(1^\lambda)$ *takes as input the security parameter and outputs a public parameter* pp.

Mark: $\mathsf{Mark}(\mathsf{pp}, C, \mathsf{m})$ *takes as input a public parameter, an arbitrary circuit $C \in \mathcal{C}_\lambda$ and a message $\mathsf{m} \in \mathcal{M}_\lambda$ and outputs a marked circuit $\widetilde{C}$.*

Extract: $\mathsf{m}' \leftarrow \mathsf{Extract}(\mathsf{pp}, C')$ *takes as input a public parameter and an arbitrary circuit C', and outputs a message m', where $\mathsf{m}' \in \mathcal{M}_\lambda \cup \{\mathsf{unmarked}\}$.*

Honest Evaluation: $\mathsf{Eval}(\mathsf{pp}, C', x)$ *takes as input a public parameter, an arbitrary circuit C', and an input x, and outputs y.*

We define the required correctness and security properties of a watermarking scheme.

Definition 4.2 (Relaxed Watermarking Property). *A watermarking scheme* $(\mathsf{Gen}, \mathsf{Mark}, \mathsf{Extract}, \mathsf{Eval})$ *for circuit family* $\{\mathcal{C}_\lambda\}_\lambda$ *and with message space* $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ *is required to satisfy the following properties.*

Statistical Correctness: *For any circuit* $C \in \mathcal{C}_\lambda$*, any message* $\mathsf{m} \in \mathcal{M}_\lambda$*, it holds that*

$$\Pr\left[\forall x\ \mathsf{Eval}(\mathsf{pp}, \widetilde{C}, x) = C(x) \;\middle|\; \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda) \\ \widetilde{C} \leftarrow \mathsf{Mark}(\mathsf{pp}, C, \mathsf{m}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Extraction Correctness: *For every* $C \in \mathcal{C}_\lambda$*,* $\mathsf{m} \in \mathcal{M}_\lambda$ *and* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$*:*

$$\Pr\left[\mathsf{m}' \neq \mathsf{m} \;\middle|\; \mathsf{m}' \leftarrow \mathsf{Extract}(\mathsf{pp}, \mathsf{Mark}(\mathsf{pp}, C, \mathsf{m}))\right] \leq \mathsf{negl}(\lambda).$$

Relaxed $(\epsilon, \mathcal{D}_\mathcal{C})$**-Unremovability:** *For every QPT* $\mathcal{A}$*, we have*

$$\Pr\left[\mathsf{Exp}^{\mathsf{r\text{-}urmv}}_{\mathcal{A},\mathcal{D}_\mathcal{C}}(\lambda, \epsilon) = 1\right] \leq \mathsf{negl}(\lambda)$$

where ϵ *is a parameter of the scheme called the* approximation factor*,* $\mathcal{D}_\mathcal{C}$ *is a distribution over* $\mathcal{C}_\lambda$*, and* $\mathsf{Exp}^{\mathsf{r\text{-}urmv}}_{\mathcal{A},\mathcal{D}_\mathcal{C}}(\lambda, \epsilon)$ *is the game defined next.*

We say a watermarking scheme is relaxed $(\epsilon, \mathcal{D}_\mathcal{C})$*-secure if it satisfies these properties.*

Definition 4.3 (Relaxed $(\epsilon, \mathcal{D}_\mathcal{C})$**-Unremovability Game).** *The game* $\mathsf{Exp}^{\mathsf{r\text{-}urmv}}_{\mathcal{A},\mathcal{D}_\mathcal{C}}(\lambda, \epsilon)$ *is defined as follows.*

1. *The challenger generates* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$ *and gives* pp *to the adversary* $\mathcal{A}$*.*
2. *At some point,* $\mathcal{A}$ *sends a message* $\mathsf{m} \in \mathcal{M}_\lambda$ *to the challenger. The challenger samples a circuit* $C \leftarrow \mathcal{D}_\mathcal{C}$ *and responds with* $\widetilde{C} \leftarrow \mathsf{Mark}(\mathsf{pp}, C, \mathsf{m})$*.*
3. *Finally, the adversary outputs a circuit* C^**. If it holds that*

$$\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{Eval}(\mathsf{pp}, C^*, x) = C(x)] \geq \epsilon$$

and $\mathsf{Extract}(\mathsf{pp}, C^*) \neq \mathsf{m}$*, then the challenger outputs* 1*, otherwise* 0*.*

Differently from the definition by Cohen et al. [CHN+18], the above definition requires a watermarking scheme has an honest evaluation algorithm for running programs. In the unremovability game above, adversaries must output a circuit whose behavior is close to the original circuit when it is executed using the honest evaluation algorithm.

Relaxed watermarking is clearly weaker than classical watermarking. However, in this work, watermarking is just an intermediate primitive, and relaxed watermarking is sufficient for our goal of constructing SSL schemes. Moreover, this relaxation allows us to achieve a public extractable watermarking scheme for a PRF family under the LWE assumption, as we will see in Section 4.2. For classical watermarking, we currently need IO to achieve such a scheme [CHN+18].

4.2 Relaxed Watermarking for PRF

We construct a relaxed watermarking scheme for PRFs from puncturable PRFs and true-simulation extractable NIZK.

Construction 4.4 (Relaxed Watermarking for PRF). Let $\mathsf{PPRF} = (\mathsf{PRF.Eval}, \mathsf{Puncture}, \mathsf{PRF.pEval})$ be a puncturable PRF whose key space, domain, and range are $\mathcal{K}$, $\{0,1\}^n$, and $\{0,1\}^m$, respectively. Also, let $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Vrfy})$ be a NIZK system for NP. Using these building blocks, we construct a relaxed watermarking scheme for the PRF family $\{\mathsf{F}_\mathsf{K}(\cdot) = \mathsf{PRF.Eval}(\mathsf{K}, \cdot) \mid \mathsf{K} \in \mathcal{K}\}$ as follows. Its message space is $\{0,1\}^k$ for some polynomial k of λ. In the construction, $\mathbf{0}$ is some fixed point in $\{0,1\}^n$.

$\mathsf{Gen}(1^\lambda)$**:** Compute $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ and Output $\mathsf{pp} := \mathsf{crs}$.

$\mathsf{Mark}(\mathsf{pp}, \mathsf{F_K}, \mathsf{m})$**:** Compute $y_{\mathbf{0}} \leftarrow \mathsf{PRF.Eval}(\mathsf{K}, \mathbf{0})$ and $\mathsf{K}_{\{\mathbf{0}\}} \leftarrow \mathsf{Puncture}(\mathsf{K}, \{\mathbf{0}\})$. Let an NP relation $\mathcal{R}_L$ be as follows.

$$\mathcal{R}_L := \left\{ \left((\mathsf{m}, y_{\mathbf{0}}, \mathsf{K}_{\{\mathbf{0}\}}), \mathsf{K} \right) \mid y_{\mathbf{0}} = \mathsf{PRF.Eval}(\mathsf{K}, \mathbf{0}), \mathsf{K}_{\{\mathbf{0}\}} = \mathsf{Puncture}(\mathsf{K}, \{\mathbf{0}\}), \text{ and } \mathsf{K} \in \mathcal{K} \right\}.$$

Compute $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{m}, y_{\mathbf{0}}, \mathsf{K}_{\{\mathbf{0}\}}), \mathsf{K})$. Output $\widetilde{C} := (\mathsf{m}, y_{\mathbf{0}}, \mathsf{K}_{\{\mathbf{0}\}}, \pi)$.

$\mathsf{Extract}(\mathsf{pp}, C')$**:** Parse $C' = (\mathsf{m}', y', K', \pi')$ and output m'.

$\mathsf{Eval}(\mathsf{pp}, C', x)$**:** Parse $C' = (\mathsf{m}', y', K', \pi')$ and run $\mathsf{NIZK.Vrfy}(\mathsf{crs}, (\mathsf{m}', y', K'), \pi)$. If the output is $\bot$, output $\bot$. Otherwise, output $\mathsf{PRF.pEval}(K', x)$ for $x \neq \mathbf{0}$ and y' for $x = \mathbf{0}$.

Theorem 4.5. *Let ϵ be any inverse polynomial of λ and $\mathcal{U}_\mathcal{K}$ the uniform distribution over $\mathcal{K}$. If* PPRF *is a puncturable PRF with key-injectiveness and* NIZK *is a true-simulation extractable NIZK system for* NP*, then Construction 4.4 is a relaxed $(\epsilon, \mathcal{U}_\mathcal{K})$-secure watermarking scheme for the PRF family $\{\mathsf{F_K}(\cdot) = \mathsf{PRF.Eval}(\mathsf{K}, \cdot) \mid \mathsf{K} \in \mathcal{K}\}$.*

Proof of Theorem 4.5. The statistical correctness of Construction 4.4 follows from the completeness of NIZK and the functionality preserving under puncturing of PPRF. Also, the extraction correctness of Construction 4.4 immediately follows from the construction. Below, we prove the relaxed $(\epsilon, \mathcal{U}_\mathcal{K})$-unremovability of Construction 4.4.

Let $\mathcal{A}$ be a QPT adversary attacking relaxed $(\epsilon, \mathcal{U}_\mathcal{K})$-unremovability. We prove this theorem using hybrid games.

Game 1: This is $\mathsf{Exp}^{\mathsf{r\text{-}urmv}}_{\mathcal{A}, \mathcal{U}_\mathcal{K}}(\lambda, \epsilon)$ for Construction 4.4.

1. The challenger generates $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ and gives $\mathsf{pp} := \mathsf{crs}$ to the adversary $\mathcal{A}$.
2. At some point, $\mathcal{A}$ queries a message $\mathsf{m} \in \{0,1\}^k$ to the challenger. The challenger first samples $\mathsf{K} \leftarrow \mathcal{U}_\mathcal{K}$. Next, the challenger computes $y_{\mathbf{0}} \leftarrow \mathsf{PRF.Eval}(\mathsf{K}, \mathbf{0})$, $\mathsf{K}_{\{\mathbf{0}\}} \leftarrow \mathsf{Puncture}(\mathsf{K}, \{\mathbf{0}\})$, and $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{m}, y_{\mathbf{0}}, \mathsf{K}_{\{\mathbf{0}\}}), \mathsf{K})$. Then, the challenger returns $\widetilde{C} := (\mathsf{m}, y_{\mathbf{0}}, \mathsf{K}_{\{\mathbf{0}\}}, \pi)$ to $\mathcal{A}$.
3. Finally, $\mathcal{A}$ outputs a circuit $C^* = (\mathsf{m}^*, y^*, \mathsf{K}^*, \pi^*)$. If $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{Eval}(\mathsf{pp}, C^*, x) = \mathsf{PRF.Eval}(\mathsf{K}, x)] \geq \epsilon$ and $\mathsf{Extract}(\mathsf{pp}, C^*) = \mathsf{m}^* \neq \mathsf{m}$ hold, then the challenger outputs 1 as the output of this game. Otherwise, the challenger outputs 0 as the output of this game.

We define the following three conditions.

(a) $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{Eval}(\mathsf{pp}, C^*, x) = \mathsf{PRF.Eval}(\mathsf{K}, x)] \geq \epsilon$.

(b) $\mathsf{NIZK.Vrfy}(\mathsf{crs}, (\mathsf{m}^*, y^*, \mathsf{K}^*), \pi^*) = \top$.

(c) $\mathsf{m}^* \neq \mathsf{m}$.

It is clear that if all of the above conditions are satisfied, the output of Game 1 is 1. In the opposite direction, it is clear that the conditions (a) and (c) are satisfied whenever the output of Game 1 is 1 from the definition of Game 1. Also, we see that if the condition (b) is not satisfied, $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{Eval}(\mathsf{pp}, C^*, x) = \mathsf{PRF.Eval}(\mathsf{K}, x)] = 0$ holds and thus the output of Game 1 is 0. Therefore, the conditions (b) is satisfied whenever the output of Game 1 is 1. Overall, the output of Game 1 is 1 if and only if the above three conditions hold in Game 1.

We define $\mathtt{S}$ as the event that the above conditions (b) and (c), and the following condition hold.

(a') Let $\omega = \lambda/\epsilon$. $\mathsf{Eval}(\mathsf{pp}, C^*, x_j) = \mathsf{PRF.Eval}(\mathsf{K}, x_j)$ holds for some $j \in [\omega]$, where x_j is randomly chosen from $\{0,1\}^n$ for every $j \in [\omega]$.

When the condition (a) is satisfied, the probability that (a') is not satisfied is bounded by $(1-\epsilon)^{\lambda/\epsilon} \leq e^{-\lambda} = \mathsf{negl}(\lambda)$. Thus, we have $\Pr[\text{Output of Game 1 is 1}] \leq \Pr[\mathtt{S}] + \mathsf{negl}(\lambda)$.

We next consider the following adversary $\mathcal{B}$ attacking the true-simulation extractability of NIZK using $\mathcal{A}$.

1. Given crs, $\mathcal{B}$ gives $\mathsf{pp} := \mathsf{crs}$ to $\mathcal{A}$.
2. When $\mathcal{A}$ queries a message $\mathsf{m} \in \{0,1\}^k$, $\mathcal{B}$ first samples $\mathsf{K} \leftarrow \mathcal{U}_{\mathcal{K}}$. Next, $\mathcal{B}$ computes $y_{\mathbf{0}} \leftarrow \mathsf{PRF.Eval}(\mathsf{K}, \mathbf{0})$ and $\mathsf{K}_{\{\mathbf{0}\}} \leftarrow \mathsf{Puncture}(\mathsf{K}, \{\mathbf{0}\})$. Then, $\mathcal{B}$ sends a statement/witness pair $((\mathsf{m}, y_{\mathbf{0}}, \mathsf{K}_{\{\mathbf{0}\}}), \mathsf{K})$ to the challenger.
3. Given π, $\mathcal{B}$ sends $\widetilde{C} := (\mathsf{m}, y_{\mathbf{0}}, \mathsf{K}_{\{\mathbf{0}\}}, \pi)$ to $\mathcal{A}$.
4. When $\mathcal{A}$ outputs $C^* = (\mathsf{m}^*, y^*, \mathsf{K}^*, \pi^*)$, $\mathcal{B}$ first randomly chooses x_j from $\{0,1\}^n$ for every $j \in [\omega]$ and checks whether $\mathsf{Eval}(\mathsf{pp}, C^*, x_j) = \mathsf{PRF.Eval}(\mathsf{K}, x_j)$ holds for some $j \in [\omega]$. If so, $\mathcal{B}$ outputs a statement/proof pair $((\mathsf{m}^*, y^*, \mathsf{K}^*), \pi^*)$. Otherwise, $\mathcal{B}$ outputs $\perp$.

When we execute $\mathsf{Expt}^{\mathsf{se\text{-}real}}_{\mathcal{B},\mathsf{NIZK}}$, the output of it is 1 if and only if the following conditions hold.

- Let $\omega = \lambda/\epsilon$. $\mathsf{Eval}(\mathsf{pp}, C^*, x_j) = \mathsf{PRF.Eval}(\mathsf{K}, x_j)$ holds for some $j \in [\omega]$, where x_j is randomly chosen from $\{0,1\}^n$ for every $j \in [\omega]$.
- $\mathsf{NIZK.Vrfy}(\mathsf{crs}, (\mathsf{m}^*, y^*, \mathsf{K}^*), \pi^*) = \top$.
- $((\mathsf{m}, y_{\mathbf{0}}, \mathsf{K}_{\{\mathbf{0}\}}), \mathsf{K}) \in \mathcal{R}_L$.
- $(\mathsf{m}, y_{\mathbf{0}}, \mathsf{K}_{\{\mathbf{0}\}}) \neq (\mathsf{m}^*, y^*, \mathsf{K}^*)$.

$\mathcal{B}$ perfectly simulates Game 1 until $\mathcal{A}$ terminates. We see that when the event $\mathtt{S}$ occurs in the simulated Game 1, the output of $\mathsf{Expt}^{\mathsf{se\text{-}real}}_{\mathcal{B},\mathsf{NIZK}}$ is 1. Namely, we have $\Pr[\mathtt{S}] \leq \Pr\left[1 \leftarrow \mathsf{Expt}^{\mathsf{se\text{-}real}}_{\mathcal{B},\mathsf{NIZK}}\right]$.

Since NIZK satisfies true-simulation extractability, there exists $\mathsf{Sim} = (\mathsf{FkSetup}, \mathsf{Sim}_1, \mathsf{Sim}_2)$ such that we have

$$\left|\Pr\left[1 \leftarrow \mathsf{Expt}^{\mathsf{se\text{-}real}}_{\mathcal{B},\mathsf{NIZK}}\right] - \Pr\left[1 \leftarrow \mathsf{Expt}^{\mathsf{se\text{-}sim}}_{\mathcal{B},\mathsf{Sim},\mathsf{NIZK}}\right]\right| \leq \mathsf{negl}(\lambda).$$

We then define the following Game 2.

Game 2: This game is the same as $\mathsf{Expt}^{\mathsf{se\text{-}sim}}_{\mathcal{B},\mathsf{Sim},\mathsf{NIZK}}$ except conceptual changes. Especially, this game is obtained by transforming $\mathsf{Expt}^{\mathsf{se\text{-}sim}}_{\mathcal{B},\mathsf{Sim},\mathsf{NIZK}}$ into a security game played between the challenger and $\mathcal{A}$ so that the output distribution does not change.

1. The challenger generates $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{FkSetup}(1^\lambda)$ and gives $\mathsf{pp} := \mathsf{crs}$ to $\mathcal{A}$.
2. When $\mathcal{A}$ queries a message $\mathsf{m} \in \{0,1\}^k$, the challenger first samples $\mathsf{K} \leftarrow \mathcal{U}_{\mathcal{K}}$. Next, the challenger computes $y_{\mathbf{0}} \leftarrow \mathsf{PRF.Eval}(\mathsf{K}, \mathbf{0})$ and $\mathsf{K}_{\{\mathbf{0}\}} \leftarrow \mathsf{Puncture}(\mathsf{K}, \{\mathbf{0}\})$. Then, the challenger computes $(\pi, \mathsf{st}_{\mathsf{Sim}}) \leftarrow \mathsf{Sim}_1(\mathsf{crs}, \mathsf{td}, (\mathsf{m}, y_{\mathbf{0}}, \mathsf{K}_{\{\mathbf{0}\}}))$ and sends $\widetilde{C} := (\mathsf{m}, y_{\mathbf{0}}, \mathsf{K}_{\{\mathbf{0}\}}, \pi)$ to $\mathcal{A}$.
3. When $\mathcal{A}$ outputs $C^* = (\mathsf{m}^*, y^*, \mathsf{K}^*, \pi^*)$, the challenger computes $\mathsf{K}' \leftarrow \mathsf{Sim}_2(\mathsf{st}_{\mathsf{Sim}}, (\mathsf{m}^*, y^*, \mathsf{K}^*), \pi^*)$. The challenger then outputs 1 if all of the following conditions hold.
   - For $\{x_j\}_{j\in[\omega]}$ randomly chosen from $\{0,1\}^n$, $\mathsf{Eval}(\mathsf{pp}, C^*, x_j) = \mathsf{PRF.Eval}(\mathsf{K}, x_j)$ holds for some $j \in [\omega]$.

- $\mathsf{NIZK.Vrfy}(\mathsf{crs}, ((\mathsf{m}^*, y^*, \mathsf{K}^*), \pi^*)) = \top$.
- $((\mathsf{m}, y_{\mathbf{0}}, \mathsf{K}_{\{\mathbf{0}\}}), \mathsf{K}) \in \mathcal{R}_L$.
- $((\mathsf{m}^*, y^*, \mathsf{K}^*), \mathsf{K}') \in \mathcal{R}_L$.
- $(\mathsf{m}, y_{\mathbf{0}}, \mathsf{K}_{\{0\}}) \neq (\mathsf{m}^*, y^*, \mathsf{K}^*)$.

Otherwise, the challenger outputs 0.

When the above first condition and fourth condition hold, we have $\mathsf{PRF.Eval}(\mathsf{K}, x_j) = \mathsf{PRF.Eval}(\mathsf{K}', x_j)$. Then, from the key-injective property of PPRF, we also have $\mathsf{K} = \mathsf{K}'$. Therefore, from the security of PPRF, we have $\Pr[\text{Output of Game 2 is 1}] \leq \mathsf{negl}(\lambda)$.

From the discussions so far, we obtain $\Pr[\text{Output of Game 1 is 1}] \leq \mathsf{negl}(\lambda)$. This completes the proof. ■

From Theorem 2.9 and Theorem 2.15, we can instantiate Construction 4.4 under the LWE assumption. Concretely, we obtain the following theorem.

Theorem 4.6. *Let ϵ be any inverse polynomial of λ. Assuming the quantum hardness of the LWE problem, there is a relaxed $(\epsilon, \mathcal{U}_{\mathsf{F}})$-secure watermarking scheme for a family of PRF $\mathcal{F}$, where $\mathcal{U}_{\mathsf{F}}$ is the uniform distribution over $\mathcal{F}$.*

4.3 Relaxed Watermarking for Compute-and-Compare Circuits

We give a construction of relaxed watermarking for circuits called (searchable) compute-and-compare circuits. The construction is essentially the classical part of the SSL construction by Ananth and La Placa [AL21]. Note that their construction uses a primitive called input-hiding obfuscation. However, our construction instead uses injective one-way functions that can be seen as a concrete instantiation of input-hiding obfuscation.

Below, we first define a family of compute-and-compare circuits and then provide the construction of a relaxed watermarking scheme for it.

Definition 4.7 (Compute-and-Compare Circuits). *A compute-and-compare circuit $\mathbf{C}\{C, \alpha\}$ is of the form*

$$\mathbf{C}\{C, \alpha\}(x) \begin{cases} 1 & (C(x) = \alpha) \\ 0 & (\textit{otherwise}) \end{cases},$$

where C is a circuit and α is a string called lock value. We let $\mathcal{C}^{n,m}_{\mathsf{cnc}} = \{\mathbf{C}\{C, \alpha\} | C : \{0,1\}^n \to \{0,1\}^m, \alpha \in \{0,1\}^m\}$.

Searchability: *We say that a family of compute-and-compare circuits $\mathcal{C}^{n,m}_{\mathsf{cnc}} = \{\mathbf{C}\{C, \alpha\} | C : \{0,1\}^n \to \{0,1\}^m, \alpha \in \{0,1\}^m\}$ is searchable if there exists a PPT algorithm $\mathcal{S}$ such that given any $\mathbf{C}\{C, \alpha\} \in \mathcal{C}^{n,m}_{\mathsf{cnc}}$, $\mathcal{S}$ outputs $x \in \{0,1\}^n$ such that $\mathbf{C}\{C, \alpha\}(x) = 1$ (i.e., $C(x) = \alpha$).*

Distribution of interest. For a function $\gamma(\lambda)$, we say that a distribution $\mathcal{D}_{\gamma\text{-}\mathsf{cnc}}$ over $\mathcal{C}^{n,m}_{\mathsf{cnc}}$ has conditional min-entropy γ if $\mathbf{C}\{C, \alpha\} \leftarrow \mathcal{D}_{\gamma\text{-}\mathsf{cnc}}$ satisfies $\mathrm{H}_\infty(\alpha | C) \geq \gamma(\lambda)$.

Construction 4.8 (Relaxed Watermarking for Searchable Compute and Compare Circuits). Let n, m, ℓ be polynomials of λ. Let $\mathcal{F}_{\mathsf{ow}} = \{f : \{0,1\}^m \to \{0,1\}^\ell\}$ be a family of injective one-way functions and let $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Vrfy})$ be a NIZK system for NP. Our relaxed watermarking scheme for searchable compute-and-compare circuits $\mathcal{C}^{n,m}_{\mathsf{cnc}}$ is as follows. Its message space is $\{0,1\}^k$ for some polynomial k of λ. Below, let $\mathcal{S}$ be the search algorithm for $\mathcal{C}^{n,m}_{\mathsf{cnc}}$.

$\mathsf{Gen}(1^\lambda)$**:** Generate $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ and $f \leftarrow \mathcal{F}_{\mathsf{ow}}$. Output $\mathsf{pp} := (\mathsf{crs}, f)$.

$\mathsf{Mark}(\mathsf{pp}, \mathbf{C}\{C,\alpha\}, \mathsf{m})$**:** Compute $x := \mathcal{S}(\mathbf{C}\{C,\alpha\})$. That is, x is an accepting point of $\mathbf{C}\{C,\alpha\}$. Compute $y \leftarrow f(\alpha)$. An NP relation $\mathcal{R}_L$ is defined as follows.

$$\mathcal{R}_L := \{((\mathsf{m}, f, y, C), x)) \mid y = f(C(x))\}.$$

Compute $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{m}, f, y, C), x)$. Output $\widetilde{C} := (\mathsf{m}, y, C, \pi)$.

$\mathsf{Extract}(\mathsf{pp}, \widetilde{C}')$**:** Parse $\widetilde{C}' = (\mathsf{m}', y', C', \pi')$ and output m'.

$\mathsf{Eval}(\mathsf{pp}, C', x)$**:** Parse $\widetilde{C}' = (\mathsf{m}', y', C', \pi')$ and run $\mathsf{NIZK.Vrfy}(\mathsf{crs}, (\mathsf{m}', f, y', C'), \pi')$. If the output is $\bot$, output $\bot$. Otherwise, output 1 if $y' = f(C'(x))$ and 0 otherwise.

Theorem 4.9. *Let n, m, and γ be functions of λ. Also, let $\mathcal{D}_{\gamma\text{-}\mathsf{cnc}}$ be any distribution over $\mathcal{C}^{n,m}_{\mathsf{cnc}}$ that has conditional min-entropy γ. If $\mathcal{F}_{\mathsf{ow}}$ is a family of injective OWF for γ-sources and NIZK is a true-simulation extractable NIZK system for NP secure against adversaries of running time $O(2^n)$, then Construction 4.8 is a relaxed $(1, \mathcal{D}_{\gamma\text{-}\mathsf{cnc}})$-secure watermarking scheme for $\mathcal{C}^{n,m}_{\mathsf{cnc}}$.*

Proof of Theorem 4.9. The statistical correctness of Construction 4.8 follows from the completeness of NIZK and the injective property of $\mathcal{F}_{\mathsf{ow}}$. Also, the extraction correctness of Construction 4.8 immediately follows from the construction. Below, we prove the relaxed $(\epsilon, \mathcal{D}_{\gamma\text{-}\mathsf{cnc}})$-unremovability of Construction 4.8.

Let $\mathcal{A}$ be a QPT adversary attacking relaxed $(1, \mathcal{D}_{\gamma\text{-}\mathsf{cnc}})$-unremovability. We prove this theorem using hybrid games.

Game 1: This is $\mathsf{Exp}^{\mathsf{r\text{-}urmv}}_{\mathcal{A},\mathcal{D}_{\gamma\text{-}\mathsf{cnc}}}(\lambda, \epsilon)$ for Construction 4.8.

1. The challenger generates $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ and $f \leftarrow \mathcal{F}_{\mathsf{ow}}$, and gives $\mathsf{pp} := (\mathsf{crs}, f)$ to the adversary $\mathcal{A}$.
2. At some point, $\mathcal{A}$ queries a message $\mathsf{m} \in \{0,1\}^k$ to the challenger. The challenger first samples $\mathbf{C}\{C,\alpha\} \leftarrow \mathcal{D}_{\gamma\text{-}\mathsf{cnc}}$. Next, the challenger computes $x := \mathcal{S}(\mathbf{C}\{C,\alpha\})$ and $y \leftarrow f(\alpha)$. Then, the challenger computes $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{m}, f, y, C), x)$. Then, the challenger returns $\widetilde{C} := (\mathsf{m}, y, C, \pi)$ to $\mathcal{A}$.
3. Finally, $\mathcal{A}$ outputs $\widetilde{C}^* = (\mathsf{m}^*, y^*, C^*, \pi^*)$. If $\mathsf{Eval}(\mathsf{pp}, \widetilde{C}^*, \cdot)$ and $\mathbf{C}\{C,\alpha\}(\cdot)$ are functionally equivalent, and $\mathsf{Extract}(\mathsf{pp}, \widetilde{C}^*) = \mathsf{m}^* \neq \mathsf{m}$, then the challenger outputs 1 as the output of this game. Otherwise, the challenger outputs 0 as the output of this game.

We define the following three conditions.

(a) $\mathsf{Eval}(\mathsf{pp}, \widetilde{C}^*, \cdot)$ and $\mathbf{C}\{C,\alpha\}(\cdot)$ are functionally equivalent.

(b) $\mathsf{NIZK.Vrfy}(\mathsf{crs}, (m^*, f, y^*, C^*), \pi^*) = \top$.

(c) $\mathsf{m}^* \neq \mathsf{m}$.

It is clear that if all of the above conditions are satisfied, the output of Game 1 is 1. In the opposite direction, it is clear that the conditions (a) and (c) are satisfied whenever the output of Game 1 is 1 from the definition of Game 1. Also, we see that if the condition (b) is not satisfied, $\mathsf{Eval}(\mathsf{pp}, \widetilde{C}^*, \cdot)$ and $\mathbf{C}\{C,\alpha\}(\cdot)$ are not functionally equivalent and thus the output of Game 1 is 0. Therefore, the condition (b) is satisfied whenever the output of Game 1 is 1. Overall, the output of Game 1 is 1 if and only if the above three conditions hold in Game 1.

We next consider the following adversary $\mathcal{B}$ attacking the true-simulation extractability of NIZK using $\mathcal{A}$.

1. Given crs, $\mathcal{B}$ generates $f \leftarrow \mathcal{F}_{\mathsf{ow}}$, and gives $\mathsf{pp} := (\mathsf{crs}, f)$ to $\mathcal{A}$.
2. When $\mathcal{A}$ queries a message $\mathsf{m} \in \{0,1\}^k$, $\mathcal{B}$ first samples $\mathbf{C}\{C, \alpha\} \leftarrow \mathcal{D}_{\gamma\text{-}\mathsf{cnc}}$. Next, $\mathcal{B}$ computes $x := \mathcal{S}(\mathbf{C}\{C, \alpha\})$ and $y \leftarrow f(\alpha)$. Then, $\mathcal{B}$ sends a statement/witness pair $((\mathsf{m}, f, y, C), x)$ to the challenger.
3. Given π, $\mathcal{B}$ sends $\widetilde{C} := (\mathsf{m}, y, C, \pi)$ to $\mathcal{A}$.
4. When $\mathcal{A}$ outputs $\widetilde{C}^* = (\mathsf{m}^*, y^*, C^*, \pi^*)$, $\mathcal{B}$ first checks whether $\mathsf{Eval}(\mathsf{pp}, C^*, \cdot)$ and $\mathbf{C}\{C, \alpha\}(\cdot)$ are functionally equivalent. (Note that this check can be done in time $O(2^n)$.) If so, $\mathcal{B}$ outputs a statement/proof pair $((m^*, f, y^*, C^*), \pi^*)$. Otherwise, $\mathcal{B}$ outputs $\perp$.

When we execute $\mathsf{Expt}^{\mathsf{se\text{-}real}}_{\mathcal{B},\mathsf{NIZK}}$, the output of it is 1 if and only if the following conditions hold.

- $\mathsf{Eval}(\mathsf{pp}, \widetilde{C}^*, \cdot)$ and $\mathbf{C}\{C, \alpha\}(\cdot)$ are functionally equivalent.
- $\mathsf{NIZK.Vrfy}(\mathsf{crs}, (m^*, f, y^*, C^*), \pi^*) = \top$.
- $((\mathsf{m}, f, y, C), x) \in \mathcal{R}_L$.
- $(\mathsf{m}, f, y, C) \neq (m^*, f, y^*, C^*)$.

$\mathcal{B}$ perfectly simulates Game 1 for $\mathcal{A}$ until $\mathcal{A}$ terminates. We see that when the output of the simulated Game 1 is 1, the output of $\mathsf{Expt}^{\mathsf{se\text{-}real}}_{\mathcal{B},\mathsf{NIZK}}$ is also 1. Namely, we have $\Pr[\text{Output of Game 1 is 1}] \leq \Pr\left[1 \leftarrow \mathsf{Expt}^{\mathsf{se\text{-}real}}_{\mathcal{B},\mathsf{NIZK}}\right]$.

$\mathcal{B}$ runs in time $O(2^n)$. Since NIZK satisfies true-simulation extractability against adversaries runs in time $O(2^n)$, there exists $\mathsf{Sim} = (\mathsf{FkSetup}, \mathsf{Sim}_1, \mathsf{Sim}_2)$ such that we have

$$\left|\Pr\left[1 \leftarrow \mathsf{Expt}^{\mathsf{se\text{-}real}}_{\mathcal{B},\mathsf{NIZK}}\right] - \Pr\left[1 \leftarrow \mathsf{Expt}^{\mathsf{se\text{-}sim}}_{\mathcal{B},\mathsf{Sim},\mathsf{NIZK}}\right]\right| \leq \mathsf{negl}(\lambda).$$

We then define the following Game 2.

Game 2: This game is the same as $\mathsf{Expt}^{\mathsf{se\text{-}sim}}_{\mathcal{B},\mathsf{Sim},\mathsf{NIZK}}$ except conceptual changes. Especially, this game is obtained by transforming $\mathsf{Expt}^{\mathsf{se\text{-}sim}}_{\mathcal{B},\mathsf{Sim},\mathsf{NIZK}}$ into a security game played between the challenger and $\mathcal{A}$ so that the output distribution does not change.

1. The challenger generates $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{FkSetup}(1^\lambda)$ and $f \leftarrow \mathcal{F}_{\mathsf{ow}}$, and gives $\mathsf{pp} := (\mathsf{crs}, f)$ to $\mathcal{A}$.
2. When $\mathcal{A}$ queries a message $\mathsf{m} \in \{0,1\}^k$, the challenger first samples $\mathbf{C}\{C, \alpha\} \leftarrow \mathcal{D}_{\gamma\text{-}\mathsf{cnc}}$. Next, the challenger computes $x := \mathcal{S}(\mathbf{C}\{C, \alpha\})$ and $y \leftarrow f(\alpha)$. Then, the challenger computes $(\pi, \mathsf{st}_{\mathsf{Sim}}) \leftarrow \mathsf{Sim}_1(\mathsf{crs}, \mathsf{td}, (\mathsf{m}, f, y, C))$ and sends $\widetilde{C} := (\mathsf{m}, y, C, \pi)$ to $\mathcal{A}$.
3. When $\mathcal{A}$ outputs $\widetilde{C}^* = (\mathsf{m}^*, y^*, \mathsf{K}^*, \pi^*)$, the challenger computes $x^* \leftarrow \mathsf{Sim}_2(\mathsf{st}_{\mathsf{Sim}}, (\mathsf{m}^*, f, y^*, C^*), \pi^*)$. The challenger then outputs 1 if all of the following conditions hold.
   - $\mathsf{Eval}(\mathsf{pp}, C^*, \cdot)$ and $\mathbf{C}\{C, \alpha\}$ are functionally equivalent.
   - $\mathsf{NIZK.Vrfy}(\mathsf{crs}, (m^*, f, y^*, C^*), \pi^*) = \top$.
   - $((\mathsf{m}, f, y, C), x) \in \mathcal{R}_L$.
   - $((m^*, f, y^*, C^*), x^*) \in \mathcal{R}_L$.
   - $(\mathsf{m}, f, y, C) \neq (m^*, f, y^*, C^*)$.

   Otherwise, the challenger outputs 0.

If the above first item and fourth item hold, we have

$$\mathbf{C}\{C,\alpha\}(x^*) = 1 \Leftrightarrow C(x^*) = \alpha,$$

and thus $f(C(x^*)) = y$. Therefore, we have $\Pr[\text{Output of Game 2 is 1}] \leq \mathsf{negl}(\lambda)$ from the security of $\mathcal{F}_{\mathsf{ow}}$.

From the discussions so far, we obtain $\Pr[\text{Output of Game 1 is 1}] \leq \mathsf{negl}(\lambda)$. This completes the proof. ■

From Theorem 2.5 and Theorem 2.15, we can instantiate Construction 4.8 under the LWE assumption. Concretely, we obtain the following theorem.

Theorem 4.10. *Let $\eta > 0$ be any constant. Assuming the hardness of the LWE problem against sub-exponential time quantum adversaries, there exists a relaxed $(1, \mathcal{D}_{\lambda^\eta\text{-}\mathsf{cnc}})$-secure watermarking scheme for the class of compute-and-compare circuits $\mathcal{C}^{n,m}_{\mathsf{cnc}}$, where $\mathcal{D}_{\lambda^\eta\text{-}\mathsf{cnc}}$ is any distribution over $\mathcal{C}^{n,m}_{\mathsf{cnc}}$ that has conditional min-entropy λ^η.*

5 Secure Software Leasing from Two-Tier Quantum Lightning

This section shows how to construct a finite-term secure SSL scheme from two-tier quantum lightning and a relaxed watermarking. Due to a technical reason, we additionally use an OT-MAC, which can be realized information theoretically.

Construction 5.1 (SSL from Two-Tier Quantum Lightning). Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ be a circuit class such that $\mathcal{C}_\lambda$ contains circuit of input length is n and output length m. Our SSL scheme $(\mathsf{Setup}, \mathsf{Gen}, \mathit{Lessor}, \mathit{Run}, \mathit{Check})$ for $\mathcal{C}$ is based on a two-tier quantum lightning $\mathsf{ttQL} = (\mathsf{ttQL.Setup}, \mathit{BoltGen}, \mathit{SemiVrfy}, \mathit{FullVrfy})$, a relaxed watermarking scheme $\mathsf{WM} = (\mathsf{WM.Gen}, \mathsf{WM.Mark}, \mathsf{WM.Extract}, \mathsf{WM.Eval})$ for $\mathcal{C}$, and a OT-MAC $\mathsf{MAC} = (\mathsf{MAC.Gen}, \mathsf{MAC.Tag}, \mathsf{MAC.Vrfy})$.

- $\mathsf{Setup}(1^\lambda)$: Compute $\mathsf{pp} \leftarrow \mathsf{WM.Gen}(1^\lambda)$ and output $\mathsf{crs} := \mathsf{pp}$.
- $\mathsf{Gen}(\mathsf{crs})$: Parse $\mathsf{pp} \leftarrow \mathsf{crs}$. Compute $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{ttQL.Setup}(1^\lambda)$ and $\mathsf{s} \leftarrow \mathsf{MAC.Gen}(1^\lambda)$, and set $\mathsf{ssl.sk} := (\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, \mathsf{s})$.
- $\mathit{Lessor}(\mathsf{ssl.sk}, C)$: Do the following:
  1. Parse $(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, \mathsf{s}) \leftarrow \mathsf{ssl.sk}$.
  2. Compute $(\mathsf{snum}, \mathit{bolt}) \leftarrow \mathit{BoltGen}(\mathsf{pk})$.
  3. Compute $\widetilde{C} \leftarrow \mathsf{WM.Mark}(\mathsf{pp}, C, \mathsf{pk}\|\mathsf{snum})$.
  4. Compute $\mathsf{tag} \leftarrow \mathsf{MAC.Tag}(\mathsf{s}, \mathsf{snum})$.
  5. Output $\mathit{sft}_C := (\mathit{bolt}, \widetilde{C}, \mathsf{tag})$.
- $\mathit{Run}(\mathsf{crs}, \mathit{sft}_C, x)$: Do the following.
  1. Parse $\mathsf{pp} \leftarrow \mathsf{crs}$ and $\mathit{sft}_C = (\mathit{bolt}, \widetilde{C}, \mathsf{tag})$.
  2. Compute $\mathsf{pk}'\|\mathsf{snum}' \leftarrow \mathsf{WM.Extract}(\mathsf{pp}, \widetilde{C})$.
  3. Run $\mathit{SemiVrfy}(\mathsf{pk}', \mathsf{snum}', \mathit{bolt})$ and obtain (b, bolt'). If $b = \bot$, then output $\bot$. Otherwise, do the next step.
  4. Compute $y \leftarrow \mathsf{WM.Eval}(\mathsf{pp}, \widetilde{C}, x)$.
  5. Output $(\mathit{bolt}', \widetilde{C}, \mathsf{tag})$ and y.

- $\mathit{Check}(\mathsf{ssl.sk}, \mathit{sft}_C)$: Do the following.
  1. Parse $(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, \mathsf{s}) \leftarrow \mathsf{ssl.sk}$ and $\mathit{sft}_C = (\mathit{bolt}, \widetilde{C}, \mathsf{tag})$.
  2. Compute $\mathsf{pk}' \| \mathsf{snum}' \leftarrow \mathsf{WM.Extract}(\mathsf{pp}, \widetilde{C})$.
  3. If $\mathsf{MAC.Vrfy}(\mathsf{s}, \mathsf{snum}', \mathsf{tag}) = \bot$, then output $\bot$. Otherwise, do the next step.
  4. Output $d \leftarrow \mathit{FullVrfy}(\mathsf{sk}, \mathsf{snum}', \mathit{bolt})$.

We have the following theorems.

Theorem 5.2. *Let ϵ be any inverse polynomial of λ and $\mathcal{D}_{\mathcal{C}}$ a distribution over $\mathcal{C}$. Assume* ttQL *is a two-tier quantum lightning scheme,* WM *is a $(\epsilon, \mathcal{D}_{\mathcal{C}})$-secure relaxed watermarking scheme for $\mathcal{C}$, and* MAC *is an OT-MAC. Then, Construction 5.1 is a $(\epsilon, \mathcal{D}_{\mathcal{C}})$-average-case finite-term lessor secure SSL scheme for $\mathcal{C}$.*

Theorem 5.3. *Let β be any inverse polynomial of λ and $\mathcal{D}_{\mathcal{C}}$ a distribution over $\mathcal{C}$. Assume* ttQL *is a two-tier quantum lightning scheme,* WM *is a $(1, \mathcal{D}_{\mathcal{C}})$-secure relaxed watermarking scheme for $\mathcal{C}$, and* MAC *is an OT-MAC. Then, Construction 5.1 is a $(\beta, \mathcal{D}_{\mathcal{C}})$-perfect finite-term lessor secure SSL scheme for $\mathcal{C}$.*

Since the proofs for the above two theorems are almost the same, we only provide the proof of Theorem 5.2 and omit the proof for Theorem 5.3.

Proof of Theorem 5.2. The correctness of $\mathcal{Run}$ of Construction 5.1 follows from the statistical correctness and extraction correctness of WM, and the semi-verification correctness of ttQL. Also, the correctness of Check of Construction 5.1 follows from the extraction correctness of WM, the correctness of MAC, and the full-verification correctness of ttQL. Below, we prove the $(\epsilon, \mathcal{D}_{\mathcal{C}})$-average-case finite-term lessor security of Construction 5.1.

Let $\mathcal{A}$ be a QPT adversary attacking $(\epsilon, \mathcal{D}_{\mathcal{C}})$-average-case finite-term lessor security. The detailed description of $\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A}, \mathcal{D}_{\mathcal{C}}}(\lambda, \epsilon)$ is as follows.

1. The challenger generates $\mathsf{pp} \leftarrow \mathsf{WM.Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{ttQL.Setup}(1^\lambda)$, and $\mathsf{s} \leftarrow \mathsf{MAC.Gen}(1^\lambda)$. The challenger then generate $C \leftarrow \mathcal{D}_{\mathcal{C}}$ and $(\mathsf{snum}, \mathit{bolt}) \leftarrow \mathit{BoltGen}(\mathsf{pk})$. The challenger also computes $\widetilde{C} \leftarrow \mathsf{WM.Mark}(\mathsf{pp}, C, \mathsf{pk} \| \mathsf{snum})$ and $\mathsf{tag} \leftarrow \mathsf{MAC.Tag}(\mathsf{s}, \mathsf{snum})$. The challenger finally sends $\mathsf{crs} := \mathsf{pp}$ and $\mathit{sft}_C := (\mathit{bolt}, \widetilde{C}, \mathsf{tag})$ to $\mathcal{A}$. Below, let $\mathsf{ssl.sk} := (\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, \mathsf{s})$.

2. $\mathcal{A}$ outputs $(\widetilde{C}^{(1)}, \mathsf{tag}^{(1)}, \widetilde{C}^{(2)}, \mathsf{tag}^{(2)}, \mathcal{B}^*)$. $(\widetilde{C}^{(1)}, \mathsf{tag}^{(1)})$ is the classical part of the first copy, and $(\widetilde{C}^{(2)}, \mathsf{tag}^{(2)})$ is that of the second copy. Moreover, $\mathcal{B}^*$ is a density matrix associated with two registers R_1 and R_2, where the states in R_1 and R_2 are associated with the first and second copy, respectively. Below, let $\mathit{sft}^{(1)} = (\mathrm{Tr}_2[\mathcal{B}^*], \widetilde{C}^{(1)}, \mathsf{tag}^{(1)})$ and $\mathit{sft}^{(2)} = (P_2(\mathsf{ssl.sk}, \mathcal{B}^*), \widetilde{C}^{(2)}, \mathsf{tag}^{(2)})$. Recall that $P_2(\mathsf{ssl.sk}, \mathcal{B}^*)$ denotes the resulting post-measurement state on R_2 after the check on R_1.

3. If it holds that $\mathit{Check}(\mathsf{ssl.sk}, \mathit{sft}^{(1)}) = \top$ and $\Pr\left[\mathcal{Run}_{\mathsf{out}}(\mathsf{crs}, \mathit{sft}^{(2)}, x) = C(x)\right] \geq \epsilon$, where the probability is taken over the choice of $x \leftarrow \{0,1\}^n$ and the random coin of $\mathcal{Run}$, then the challenger outputs 1 as the output of this game. Otherwise, the challenger outputs 0 as the output of this game.

Below, we let $\mathsf{pk}^{(1)} \| \mathsf{snum}^{(1)} \leftarrow \mathsf{WM.Extract}(\mathsf{pp}, \widetilde{C}^{(1)})$ and $\mathsf{pk}^{(2)} \| \mathsf{snum}^{(2)} \leftarrow \mathsf{WM.Extract}(\mathsf{pp}, \widetilde{C}^{(2)})$. The output of $\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A}, \mathcal{D}_{\mathcal{C}}}(\lambda, \epsilon)$ is 1 if and only if the following conditions hold.

(a) $\mathsf{MAC.Vrfy}(\mathsf{s}, \mathsf{snum}^{(1)}, \mathsf{tag}^{(1)}) = \top$.

(b) $\mathit{FullVrfy}(\mathsf{sk}, \mathsf{snum}^{(1)}, \mathrm{Tr}_2[\mathcal{B}^*]) = \top$.

(c) $\mathit{SemiVrfy}(\mathsf{pk}^{(2)}, \mathsf{snum}^{(2)}, P_2(\mathsf{ssl.sk}, b^*)) = \top$.

(d) $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{WM.Eval}(\mathsf{crs}, \widetilde{C}^{(2)}, x) = C(x)] \geq \epsilon$.

We can estimate the advantage of $\mathcal{A}$ as

$$\begin{aligned}
\Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_\mathcal{C}}(\lambda,\epsilon) = 1\right] &= \Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_\mathcal{C}}(\lambda,\epsilon) = 1 \wedge \mathsf{snum}^{(1)} = \mathsf{snum} \wedge \mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} = \mathsf{pk}\|\mathsf{snum}\right] \\
&\quad + \Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_\mathcal{C}}(\lambda,\epsilon) = 1 \wedge (\mathsf{snum}^{(1)} \neq \mathsf{snum} \vee \mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} \neq \mathsf{pk}\|\mathsf{snum})\right] \\
&\leq \Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_\mathcal{C}}(\lambda,\epsilon) = 1 \wedge \mathsf{snum}^{(1)} = \mathsf{snum} \wedge \mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} = \mathsf{pk}\|\mathsf{snum}\right] \\
&\quad + \Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_\mathcal{C}}(\lambda,\epsilon) = 1 \wedge \mathsf{snum}^{(1)} \neq \mathsf{snum}\right] \\
&\quad + \Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_\mathcal{C}}(\lambda,\epsilon) = 1 \wedge \mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} \neq \mathsf{pk}\|\mathsf{snum}\right]
\end{aligned}$$

We then have the following lemmas.

Lemma 5.4. $\Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_\mathcal{C}}(\lambda,\epsilon) = 1 \wedge \mathsf{snum}^{(1)} = \mathsf{snum} \wedge \mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} = \mathsf{pk}\|\mathsf{snum}\right] = \mathsf{negl}(\lambda)$ *by the two-tier unclonability of* ttQL.

Lemma 5.5. $\Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_\mathcal{C}}(\lambda,\epsilon) = 1 \wedge \mathsf{snum}^{(1)} \neq \mathsf{snum}\right] = \mathsf{negl}(\lambda)$ *by the security of* MAC.

Lemma 5.6. $\Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_\mathcal{C}}(\lambda,\epsilon) = 1 \wedge \mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} \neq \mathsf{pk}\|\mathsf{snum}\right] = \mathsf{negl}(\lambda)$ *by the* $(\epsilon, \mathcal{D}_\mathcal{C})$*-removability of* WM.

For Lemma 5.4, if the condition (b) and (c) above and $\mathsf{snum}^{(1)} = \mathsf{snum} \wedge \mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} = \mathsf{pk}\|\mathsf{snum}$ hold at the same time with non-negligible probability, by using $\mathcal{A}$, we can construct an adversary breaking the two-tier unclonability of ttQL. Thus, we have Lemma 5.4. Next, for Lemma 5.5, if the condition (a) and $\mathsf{snum}^{(1)} \neq \mathsf{snum}$ hold with non-negligible probability, also by using $\mathcal{A}$, we can construct an adversary breaking the security of MAC. Thus, we have Lemma 5.5. Finally, for Lemma 5.6, if the condition (d) and $\mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} \neq \mathsf{pk}\|\mathsf{snum}$ hold with non-negligible probability, by using $\mathcal{A}$, we can construct an adversary breaking $(\epsilon, \mathcal{D}_\mathcal{C})$-unremovability of WM. Thus, we have Lemma 5.6.

From the discussions so far, we obtain $\Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_\mathcal{C}}(\lambda,\epsilon) = 1\right] \leq \mathsf{negl}(\lambda)$. This completes the proof. ■

6 Secure Software Leasing with Classical Communication

In this section, we extend our finite-term secure SSL scheme to one with classical communication by using two-tier quantum lightning with classical verification.

6.1 Definition

First, we formalize the notion of SSL with classical communication.

Definition 6.1 (SSL with Setup and classical communication). *Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ be a circuit class such that $\mathcal{C}_\lambda$ contains circuit of input length is n and output length m. A secure software lease scheme with setup and classical communication for $\mathcal{C}$ is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Lessor}, \mathcal{L}\mathit{essee}_1, \mathcal{L}\mathit{essee}_2, \mathcal{R}\mathit{un}, \mathcal{SSLC}\mathit{ert}, \mathsf{CertVrfy})$.

- $\mathsf{Setup}(1^\lambda), \mathsf{Gen}(\mathsf{crs}), \mathcal{R}\mathit{un}(\mathsf{crs}, \mathit{sft}_\mathsf{C}, x)$*: These are the same as the SSL in Definition 2.18.*

- $\mathsf{Gen}(\mathsf{crs})$: *The key generation algorithm takes as input* crs *and outputs a public key* pk *and secret key* sk.
- $\mathcal{L}\mathit{essee}_1(\mathsf{pk})$: *The first stage lessee algorithm takes as input* crs *and outputs a classical string* obligation *and a quantum state* $\mathit{st}_{\mathcal{L}\mathit{essee}}$.
- $\mathsf{Lessor}(\mathsf{sk}, \mathsf{obligation}, C)$: *The lessor algorithm takes as input* sk, obligation, *and a circuit* $C \in \mathcal{C}$, *and outputs a classical string* answer.
- $\mathcal{L}\mathit{essee}_2(\mathsf{pk}, \mathit{st}_{\mathcal{L}\mathit{essee}}, \mathsf{answer})$: *The second stage lessee algorithm takes as input* crs, $\mathit{st}_{\mathcal{L}\mathit{essee}}$, *and* answer, *and outputs a quantum state* sft.
- $\mathcal{SSLC}\mathit{ert}(\mathsf{crs}, \mathit{sft}^*)$: *The certification algorithm takes as input* crs *and* sft^* *and outputs a classical string* cert.
- $\mathsf{CertVrfy}(\mathsf{sk}, \mathsf{cert})$: *The certification-verification algorithm takes as input* sk *and* cert *and outputs* $\top$ *or* $\bot$.

Definition 6.2 (Correctness for SSL with classical verification). *An SSL scheme with classical communication* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Lessor}, \mathcal{L}\mathit{essee}_1, \mathcal{L}\mathit{essee}_2, \mathcal{R}\mathit{un}, \mathcal{C}\mathit{heck})$ *for* $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ *is correct if for all* $C \in \mathcal{C}_\lambda$, *the following two properties hold:*

- *Correctness of* $\mathcal{R}\mathit{un}$*:*

$$\Pr\left[\begin{array}{l|l} \forall x, \Pr\left[\mathcal{R}\mathit{un}_{\mathsf{out}}(\mathsf{crs}, \mathit{sft}_C, x) = C(x)\right] \geq 1 - \mathsf{negl}(\lambda) & \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{crs}) \\ (\mathsf{obligation}, \mathit{st}_{\mathcal{L}\mathit{essee}}) \leftarrow \mathcal{L}\mathit{essee}_1(\mathsf{pk}) \\ \mathsf{answer} \leftarrow \mathsf{Lessor}(\mathsf{sk}, \mathsf{obligation}, C) \\ \mathit{sft}_C \leftarrow \mathcal{L}\mathit{essee}_2(\mathsf{pk}, \mathit{st}_{\mathcal{L}\mathit{essee}}, \mathsf{answer}) \end{array} \end{array}\right]$$

$$\geq 1 - \mathsf{negl}(\lambda).$$

- *Correctness of* CertVrfy*:*

$$\Pr\left[\mathsf{CertVrfy}(\mathsf{sk}, \mathsf{cert}) = \top \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{crs}) \\ (\mathsf{obligation}, \mathit{st}_{\mathcal{L}\mathit{essee}}) \leftarrow \mathcal{L}\mathit{essee}_1(\mathsf{pk}) \\ \mathsf{answer} \leftarrow \mathsf{Lessor}(\mathsf{sk}, \mathsf{obligation}, C) \\ \mathit{sft}_C \leftarrow \mathcal{L}\mathit{essee}_2(\mathsf{pk}, \mathit{st}_{\mathcal{L}\mathit{essee}}, \mathsf{answer}) \\ \mathsf{cert} \leftarrow \mathcal{SSLC}\mathit{ert}(\mathsf{crs}, \mathit{sft}_C) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Similarly to the ordinary SSL, we consider the following two security notions perfect finite-term lessor security and average-case finite-term lessor security.

Definition 6.3 (Perfect Finite-Term Lessor Security). *Let* β *be any inverse polynomial of* λ *and* $\mathcal{D}_\mathcal{C}$ *a distribution on* $\mathcal{C}$. *We define the* $(\beta, \mathcal{D}_\mathcal{C})$*-perfect finite-term lessor security game* $\mathsf{Expt}^{\mathsf{pft\text{-}lessor\text{-}cc}}_{\mathcal{A}, \mathcal{D}_\mathcal{C}}(\lambda, \beta)$ *between the challenger and adversary* $\mathcal{A}$ *as follows.*

1. *The challenger generates* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{crs})$, *and sends* crs *and* pk *to* $\mathcal{A}$.
2. $\mathcal{A}$ *outputs* obligation. *The challenger generates* $C \leftarrow \mathcal{D}_\mathcal{C}$, *computes* $\mathsf{answer} \leftarrow \mathsf{Lessor}(\mathsf{sk}, \mathsf{obligation}, C)$, *and sends* answer *to* $\mathcal{A}$.
3. $\mathcal{A}$ *outputs a classical string* cert^* *and a quantum state* sft^*.

4. *If* $\mathsf{CertVrfy}(\mathsf{sk}, \mathsf{cert}^*) = \top$ *and* $\forall x\ \Pr[\mathcal{R}un_{\mathsf{out}}(\mathsf{crs}, \mathit{sft}^*, x) = C(x)] \geq \beta$ *hold, where the probability is taken over the choice of the random coin of* $\mathcal{R}un$*, then the challenger outputs* 1*. Otherwise, the challenger outputs* 0*.*

We say that an SSL scheme with classical communication $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Lessor}, \mathcal{L}essee_1, \mathcal{L}essee_2, \mathcal{R}un, \mathcal{SSLC}ert, \mathsf{CertVrfy})$ *is* $(\beta, \mathcal{D}_{\mathcal{C}})$*-perfect finite-term lessor secure, if for any QPT* $\mathcal{A}$*, the following holds.*

$$\Pr\left[\mathsf{Expt}^{\mathsf{pft\text{-}lessor\text{-}cc}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda, \beta) = 1\right] \leq \mathsf{negl}(\lambda).$$

Definition 6.4 (Average-Case Finite-Term Lessor Security). *Let* ϵ *be any inverse polynomial of* λ *and* $\mathcal{D}_{\mathcal{C}}$ *a distribution on* $\mathcal{C}$*. We define the* $(\epsilon, \mathcal{D}_{\mathcal{C}})$*-average-case finite-term lessor security game* $\mathsf{Expt}^{\mathsf{aft\text{-}lessor\text{-}cc}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda, \epsilon)$ *by replacing the fourth stage of* $\mathsf{Expt}^{\mathsf{pft\text{-}lessor\text{-}cc}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda, \beta)$ *with the following.*

4. *If* $\mathsf{CertVrfy}(\mathsf{sk}, \mathsf{cert}^*) = \top$ *and* $\Pr[\mathcal{R}un_{\mathsf{out}}(\mathsf{crs}, \mathit{sft}^*, x) = C(x)] \geq \epsilon$ *hold, where the probability is taken over the choice of* $x \leftarrow \{0,1\}^n$ *and the random coin of* $\mathcal{R}un$*, then the challenger outputs* 1*. Otherwise, the challenger outputs* 0*.*

We say that an SSL scheme with classical communication $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Lessor}, \mathcal{L}essee_1, \mathcal{L}essee_2, \mathcal{R}un, \mathcal{SSLC}ert, \mathsf{CertVrfy})$ *is* $(\epsilon, \mathcal{D}_{\mathcal{C}})$*-average-case finite-term lessor secure, if for any QPT* $\mathcal{A}$*, the following holds.*

$$\Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda, \epsilon) = 1\right] \leq \mathsf{negl}(\lambda).$$

6.2 Construction

We show how to construct a finite-term secure SSL scheme with classical communication from two-tier quantum lightning with classical verification, relaxed watermarking, and OT-MAC.

Construction 6.5 (SSL from Two-Tier QL with classical verification). Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ be a circuit class such that $\mathcal{C}_\lambda$ contains circuit of input length is n and output length m. Our SSL scheme with classical communication $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Lessor}, \mathcal{L}essee_1, \mathcal{L}essee_2, \mathcal{R}un, \mathcal{SSLC}ert, \mathsf{CertVrfy})$ for $\mathcal{C}$ is based on a two-tier QL with semi-classical verification $\mathsf{ttQL} = (\mathsf{ttQL.Setup}, \mathcal{B}oltGen, \mathcal{B}oltCert, \mathcal{S}emiVrfy, \mathsf{ttQL.CertVrfy})$, relaxed watermarking scheme $\mathsf{WM} = (\mathsf{WM.Setup}, \mathsf{WM.Mark}, \mathsf{WM.Extract}, \mathsf{WM.Eval})$ for $\mathcal{C}$, and OT-MAC $\mathsf{MAC} = (\mathsf{MAC.Gen}, \mathsf{MAC.Tag}, \mathsf{MAC.Vrfy})$.

- $\mathsf{Setup}(1^\lambda)$: Compute $\mathsf{pp} \leftarrow \mathsf{WM.Gen}(1^\lambda)$ and output $\mathsf{crs} := \mathsf{pp}$.
- $\mathsf{Gen}(\mathsf{crs})$: Compute $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{ttQL.Setup}(1^\lambda)$ and $\mathsf{s} \leftarrow \mathsf{MAC.Gen}(1^\lambda)$, and output $\mathsf{ssl.pk} := \mathsf{pk}$ and $\mathsf{ssl.sk} := (\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, \mathsf{s})$.
- $\mathcal{L}essee_1(\mathsf{ssl.pk})$: Parse $\mathsf{pk} \leftarrow \mathsf{ssl.pk}$, generate $(\mathsf{snum}, \mathit{bolt}) \leftarrow \mathcal{B}oltGen(\mathsf{pk})$, and outputs $\mathsf{obligation} := \mathsf{snum}$ and $\mathit{st}_{\mathcal{L}essee} := \mathit{bolt}$.
- $\mathsf{Lessor}(\mathsf{sk}, \mathsf{obligation}, C)$:
  1. Parse $(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, \mathsf{s}) \leftarrow \mathsf{ssl.sk}$ and $\mathsf{snum} \leftarrow \mathsf{obligation}$.
  2. Compute $\widetilde{C} \leftarrow \mathsf{WM.Mark}(\mathsf{pp}, C, \mathsf{pk} \| \mathsf{snum})$.
  3. Compute $\mathsf{tag} \leftarrow \mathsf{MAC.Tag}(\mathsf{s}, \mathsf{snum})$.
  4. Output $\mathsf{answer} := (\widetilde{C}, \mathsf{tag})$.
- $\mathcal{L}essee_2(\mathsf{ssl.pk}, \mathit{st}_{\mathcal{L}essee}, \mathsf{answer})$: Parse $\mathit{bolt} \leftarrow \mathit{st}_{\mathcal{L}essee}$ and $(\widetilde{C}, \mathsf{tag}) \leftarrow \mathsf{answer}$, and output $\mathit{sft} := (\mathit{bolt}, \widetilde{C}, \mathsf{tag})$.

- $\mathcal{Run}(\mathsf{crs}, \mathsf{sft}_C, x)$: Do the following.
  1. Parse $\mathsf{pp} \leftarrow \mathsf{crs}$ and $(\mathit{bolt}, \widetilde{C}, \mathsf{tag}) \leftarrow \mathit{sft}$.
  2. Compute $\mathsf{pk}' \| \mathsf{snum}' \leftarrow \mathsf{WM.Extract}(\mathsf{pp}, \widetilde{C})$.
  3. Run $(b, \mathit{bolt}') \leftarrow \mathit{SemiVrfy}(\mathsf{pk}', \mathsf{snum}', \mathit{bolt})$. If $b = \bot$, then output $\bot$. Otherwise, do the next step.
  4. Compute $y \leftarrow \mathsf{WM.Eval}(\mathsf{pp}, \widetilde{C}, x)$.
  5. Output $(\mathit{bolt}', \widetilde{C}, \mathsf{tag})$ and y.
- $\mathit{SSLCert}(\mathsf{crs}, \mathit{sft})$: Parse $(\mathit{bolt}, \widetilde{C}, \mathsf{tag}) \leftarrow \mathit{sft}$, runs $\mathsf{ttQL.cert} \leftarrow \mathit{BoltCert}(\mathit{bolt})$, and output $\mathsf{cert} := (\mathsf{ttQL.cert}, \widetilde{C}, \mathsf{tag})$.
- $\mathsf{CertVrfy}(\mathsf{ssl.sk}, \mathsf{cert})$: Do the following.
  1. Parse $(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, \mathsf{s}) \leftarrow \mathsf{ssl.sk}$ and $(\mathsf{ttQL.cert}, \widetilde{C}, \mathsf{tag}) \leftarrow \mathsf{cert}$.
  2. Compute $\mathsf{pk}' \| \mathsf{snum}' \leftarrow \mathsf{WM.Extract}(\mathsf{pp}, \widetilde{C})$.
  3. If $\mathsf{MAC.Vrfy}(\mathsf{s}, \mathsf{snum}', \mathsf{tag}) = \bot$, then output $\bot$. Otherwise, do the next step.
  4. Output $d \leftarrow \mathsf{ttQL.CertVrfy}(\mathsf{sk}, \mathsf{snum}', \mathsf{cert})$.

We have the following theorems.

Theorem 6.6. *Let ϵ be any inverse polynomial of λ and $\mathcal{D}_{\mathcal{C}}$ a distribution over $\mathcal{C}$. Assume* ttQL *is a two-tier quantum lightning scheme with classical verification,* WM *is a $(\epsilon, \mathcal{D}_{\mathcal{C}})$-secure relaxed watermarking scheme for $\mathcal{C}$, and* MAC *is an OT-MAC. Then, Construction 6.5 is a $(\epsilon, \mathcal{D}_{\mathcal{C}})$-average-case finite-term lessor secure SSL scheme with classical communication for $\mathcal{C}$.*

Theorem 6.7. *Let β be any inverse polynomial of λ and $\mathcal{D}_{\mathcal{C}}$ a distribution over $\mathcal{C}$. Assume* ttQL *is a two-tier quantum lightning scheme with classical verification,* WM *is a $(1, \mathcal{D}_{\mathcal{C}})$-secure relaxed watermarking scheme for $\mathcal{C}$, and* MAC *is an OT-MAC. Then, Construction 6.5 is a $(\beta, \mathcal{D}_{\mathcal{C}})$-perfect finite-term lessor secure SSL scheme with classical communication for $\mathcal{C}$.*

Since the proofs for the above two theorems are almost the same, we provide the proof of only Theorem 6.6 and omit the proof of Theorem 6.7.

Proof of Theorem 6.6. The correctness of $\mathcal{Run}$ of Construction 6.5 follows from the statistical correctness and extraction correctness of WM, and the semi-verification correctness of ttQL. Also, the correctness of CertVrfy of Construction 6.5 follows from the extraction correctness of WM, the correctness of MAC, and the certification-verification correctness of ttQL. Below, we prove the $(\epsilon, \mathcal{D}_{\mathcal{C}})$-average-case finite-term lessor security of Construction 6.5.

Let $\mathcal{A}$ be a QPT adversary attacking $(\epsilon, \mathcal{D}_{\mathcal{C}})$-average-case finite-term lessor security. The detailed description of $\mathsf{Expt}^{\mathsf{aft\text{-}lessor\text{-}cc}}_{\mathcal{A}, \mathcal{D}_{\mathcal{C}}}(\lambda, \epsilon)$ is as follows.

1. The challenger generates $\mathsf{pp} \leftarrow \mathsf{WM.Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{ttQL.Setup}(1^\lambda)$, and $\mathsf{s} \leftarrow \mathsf{MAC.Gen}(1^\lambda)$. The challenger sends $\mathsf{crs} := \mathsf{pp}$ and $\mathsf{ssl.pk} := \mathsf{pk}$ to $\mathcal{A}$. Below, let $\mathsf{ssl.sk} := (\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, \mathsf{s})$.
2. $\mathcal{A}$ sends $\mathsf{obligation} := \mathsf{snum}^*$ to the challenger. The challenger generates $C \leftarrow \mathcal{D}_{\mathcal{C}}$. The challenger also computes $\widetilde{C} \leftarrow \mathsf{WM.Mark}(\mathsf{pp}, C, \mathsf{pk} \| \mathsf{snum}^*)$ and $\mathsf{tag} \leftarrow \mathsf{MAC.Tag}(\mathsf{s}, \mathsf{snum}^*)$. The challenger finally sends $\mathsf{answer} := (\widetilde{C}, \mathsf{tag})$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs $\mathsf{cert}^* = (\mathsf{ttQL.cert}^*, \widetilde{C}^{(1)}, \mathsf{tag}^{(1)})$ and $\mathit{sft}^* = (\mathit{bolt}^*, \widetilde{C}^{(2)}, \mathsf{tag}^{(2)})$, where bolt^* is a single quantum state and others are classical strings.

4. If it holds that $\mathsf{CertVrfy}(\mathsf{ssl.sk}, \mathsf{cert}) = \top$ and $\Pr[\mathcal{Run}_{\mathsf{out}}(\mathsf{crs}, \mathit{sft}^*, x) = C(x)] \geq \epsilon$, where the probability is taken over the choice of $x \leftarrow \{0,1\}^n$ and the random coin of $\mathcal{Run}$, then the challenger outputs 1 as the output of this game. Otherwise, the challenger outputs 0 as the output of this game.

Below, we let $\mathsf{pk}^{(1)} \| \mathsf{snum}^{(1)} \leftarrow \mathsf{WM.Extract}(\mathsf{pp}, \widetilde{C}^{(1)})$ and $\mathsf{pk}^{(2)} \| \mathsf{snum}^{(2)} \leftarrow \mathsf{WM.Extract}(\mathsf{pp}, \widetilde{C}^{(2)})$. The output of $\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda, \epsilon)$ is 1 if and only if the following conditions hold.

(a) $\mathsf{MAC.Vrfy}(\mathsf{s}, \mathsf{snum}^{(1)}, \mathsf{tag}^{(1)}) = \top$.

(b) $\mathsf{ttQL.CertVrfy}(\mathsf{sk}, \mathsf{snum}^{(1)}, \mathsf{ttQL.cert}^*) = \top$.

(c) $\mathit{SemiVrfy}(\mathsf{pk}^{(2)}, \mathsf{snum}^{(2)}, \mathit{b}^*) = \top$.

(d) $\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{WM.Eval}(\mathsf{pp}, \widetilde{C}^{(2)}, x) = C(x)] \geq \epsilon$.

We can estimate the advantage of $\mathcal{A}$ as

$$\begin{aligned}
\Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda,\epsilon) = 1\right] &= \Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda,\epsilon) = 1 \wedge \mathsf{snum}^{(1)} = \mathsf{snum}^* \wedge \mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} = \mathsf{pk}\|\mathsf{snum}^*\right] \\
&\quad + \Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda,\epsilon) = 1 \wedge (\mathsf{snum}^{(1)} \neq \mathsf{snum}^* \vee \mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} \neq \mathsf{pk}\|\mathsf{snum}^*)\right] \\
&\leq \Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda,\epsilon) = 1 \wedge \mathsf{snum}^{(1)} = \mathsf{snum}^* \wedge \mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} = \mathsf{pk}\|\mathsf{snum}^*\right] \\
&\quad + \Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda,\epsilon) = 1 \wedge \mathsf{snum}^{(1)} \neq \mathsf{snum}^*\right] \\
&\quad + \Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda,\epsilon) = 1 \wedge \mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} \neq \mathsf{pk}\|\mathsf{snum}^*\right]
\end{aligned}$$

We then have the following lemmas.

Lemma 6.8. $\Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda,\epsilon) = 1 \wedge \mathsf{snum}^{(1)} = \mathsf{snum}^* \wedge \mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} = \mathsf{pk}\|\mathsf{snum}^*\right] = \mathsf{negl}(\lambda)$ *by the two-tier unclonability with classical verification of* ttQL.

Lemma 6.9. $\Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda,\epsilon) = 1 \wedge \mathsf{snum}^{(1)} \neq \mathsf{snum}^*\right] = \mathsf{negl}(\lambda)$ *by the security of* MAC.

Lemma 6.10. $\Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda,\epsilon) = 1 \wedge \mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} \neq \mathsf{pk}\|\mathsf{snum}^*\right] = \mathsf{negl}(\lambda)$ *by the* $(\epsilon, \mathcal{D}_{\mathcal{C}})$*-unremovability of* WM.

For Lemma 6.8, if the condition (b) and (c) above and $\mathsf{snum}^{(1)} = \mathsf{snum}^* \wedge \mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} = \mathsf{pk}\|\mathsf{snum}^*$ hold at the same time with non-negligible probability, by using $\mathcal{A}$, we can construct an adversary breaking the two-tier unclonability of ttQL. Thus, we have Lemma 6.8. Next, for Lemma 6.9, if the condition (a) and $\mathsf{snum}^{(1)} \neq \mathsf{snum}^*$ hold with non-negligible probability, also by using $\mathcal{A}$, we can construct an adversary breaking the security of MAC. Thus, we have Lemma 6.9. Finally, for Lemma 6.10, if the condition (d) and $\mathsf{pk}^{(2)}\|\mathsf{snum}^{(2)} \neq \mathsf{pk}\|\mathsf{snum}^*$ hold with non-negligible probability, by using $\mathcal{A}$, we can construct an adversary breaking $(\epsilon, \mathcal{D}_{\mathcal{C}})$-unremovability of WM. Thus, we have Lemma 6.10.

From the discussions so far, we obtain $\Pr\left[\mathsf{Expt}^{\mathsf{aft\text{-}lessor}}_{\mathcal{A},\mathcal{D}_{\mathcal{C}}}(\lambda,\epsilon) = 1\right] \leq \mathsf{negl}(\lambda)$. This completes the proof. ■

7 Putting It Altogether: SSL from LWE

In this section, we summarize our results.

SSL for a family of PRF. By combining Theorem 5.2 with Theorem 3.16, Corollary 3.21, Theorem 4.6, and Theorem 2.11, we obtain the following theorem.

Theorem 7.1. *Let ϵ be any inverse polynomial of λ. Assuming the quantum hardness of the LWE problem, there exists a $(\epsilon, \mathcal{U}_{\mathsf{F}})$-average-case finite-term lessor secure SSL scheme for a family of PRF $\mathcal{F}$, where $\mathcal{U}_{\mathsf{F}}$ is the uniform distribution over $\mathcal{F}$.*

Also, by combing Theorem 6.6 with Corollary 3.21, Theorem 4.6, and Theorem 2.11, we obtain the following theorem.

Theorem 7.2. *Let ϵ be any inverse polynomial of λ. Assuming the quantum hardness of the LWE problem, there exists a $(\epsilon, \mathcal{U}_{\mathsf{F}})$-average-case finite-term lessor secure SSL scheme with classical communication for a family of PRF $\mathcal{F}$, where $\mathcal{U}_{\mathsf{F}}$ is the uniform distribution over $\mathcal{F}$.*

SSL for compute-and-compare circuits. By combining Theorem 5.3 with Theorem 3.16, Corollary 3.21, Theorem 4.10, and Theorem 2.11, we obtain the following theorem.

Theorem 7.3. *Let β be any inverse polynomial of λ and $\eta > 0$ any constant. Assuming the hardness of the LWE problem against sub-exponential time quantum adversaries, there exists a $(\beta, \mathcal{D}_{\lambda^\eta\text{-}\mathsf{cnc}})$-perfect finite-term lessor secure SSL scheme for the class of compute-and-compare circuits $\mathcal{C}^{n,m}_{\mathsf{cnc}}$, where $\mathcal{D}_{\lambda^\eta\text{-}\mathsf{cnc}}$ is any distribution over $\mathcal{C}^{n,m}_{\mathsf{cnc}}$ that has conditional min-entropy λ^η.*

Also, by combing Theorem 6.7 with Corollary 3.21, Theorem 4.10, and Theorem 2.11, we obtain the following theorem.

Theorem 7.4. *Let β be any inverse polynomial of λ and $\eta > 0$ any constant. Assuming the hardness of the LWE problem against sub-exponential time quantum adversaries, there exists a $(\beta, \mathcal{D}_{\lambda^\eta\text{-}\mathsf{cnc}})$-perfect finite-term lessor secure SSL scheme with classical communication for the class of compute-and-compare circuits $\mathcal{C}^{n,m}_{\mathsf{cnc}}$, where $\mathcal{D}_{\lambda^\eta\text{-}\mathsf{cnc}}$ is any distribution over $\mathcal{C}^{n,m}_{\mathsf{cnc}}$ that has conditional min-entropy λ^η.*

References

[Aar05] Scott Aaronson. Limitations of quantum advice and one-way communication. *Theory of Computing*, 1(1):1–28, 2005. (Cited on page 18.)

[Aar09] Scott Aaronson. Quantum copy-protection and quantum money. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009*, pages 229–242. IEEE Computer Society, 2009. (Cited on page 1.)

[AC12] Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 41–60. ACM Press, May 2012. (Cited on page 2, 4, 6.)

[AGKZ20] Ryan Amos, Marios Georgiou, Aggelos Kiayias, and Mark Zhandry. One-shot signatures and applications to hybrid quantum/classical authentication. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *52nd ACM STOC*, pages 255–268. ACM Press, June 2020. (Cited on page 2, 3, 4.)

[AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 57–74. Springer, Heidelberg, August 2013. (Cited on page 11.)

[AL21] Prabhanjan Ananth and Rolando L. La Placa. Secure software leasing. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 501–530. Springer, Heidelberg, October 2021. (Cited on page 1, 2, 3, 4, 5, 7, 8, 13, 15, 16, 18, 27.)

[ALL+21] Scott Aaronson, Jiahui Liu, Qipeng Liu, Mark Zhandry, and Ruizhe Zhang. New approaches for quantum copy-protection. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 526–555, Virtual Event, August 2021. Springer, Heidelberg. (Cited on page 3.)

[ALZ20] Scott Aaronson, Jiahui Liu, and Ruizhe Zhang. Quantum copy-protection from hidden subspaces. *CoRR*, abs/2004.09674, 2020. version v5 or older. (Cited on page 1, 3.)

[AP20] Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 110–140. Springer, Heidelberg, May 2020. (Cited on page 2.)

[BCM+18] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In Mikkel Thorup, editor, *59th FOCS*, pages 320–331. IEEE Computer Society Press, October 2018. (Cited on page 7, 9, 14, 15.)

[BDGM20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for iO: Circular-secure LWE suffices. Cryptology ePrint Archive, Report 2020/1024, 2020. https://eprint.iacr.org/2020/1024. (Cited on page 2.)

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6:1–6:48, 2012. (Cited on page 2, 3.)

[BGMZ18] James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Return of GGH15: Provable security against zeroizing attacks. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 544–574. Springer, Heidelberg, November 2018. (Cited on page 2.)

[BJL+21] Anne Broadbent, Stacey Jeffery, Sébastien Lord, Supartha Podder, and Aarthi Sundaram. Secure software leasing without assumptions, 2021. (Cited on page 3.)

[CHN+18] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. *SIAM Journal on Computing*, 47(6):2157–2202, 2018. (Cited on page 3, 12, 23, 24.)

[CHS05] Ran Canetti, Shai Halevi, and Michael Steiner. Hardness amplification of weakly verifiable puzzles. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 17–33. Springer, Heidelberg, February 2005. (Cited on page 9, 21.)

[CHVW19] Yilei Chen, Minki Hhan, Vinod Vaikuntanathan, and Hoeteck Wee. Matrix PRFs: Constructions, attacks, and applications to obfuscation. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 55–80. Springer, Heidelberg, December 2019. (Cited on page 2.)

[CMP20] Andrea Coladangelo, Christian Majenz, and Alexander Poremba. Quantum copy-protection of compute-and-compare programs in the quantum random oracle model, 2020. (Cited on page 3.)

[CS20] Andrea Coladangelo and Or Sattath. A quantum money solution to the blockchain scalability problem. *CoRR*, abs/2002.11998, 2020. (Cited on page 9, 19.)

[FGH+12] Edward Farhi, David Gosset, Avinatan Hassidim, Andrew Lutomirski, and Peter W. Shor. Quantum money from knots. In Shafi Goldwasser, editor, *ITCS 2012*, pages 276–289. ACM, January 2012. (Cited on page 2.)

[GKM+19] Rishab Goyal, Sam Kim, Nathan Manohar, Brent Waters, and David J. Wu. Watermarking public-key cryptographic primitives. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 367–398. Springer, Heidelberg, August 2019. (Cited on page 3.)

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 736–749. ACM, 2021. (Cited on page 2.)

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. (Cited on page 18.)

[GZ20] Marios Georgiou and Mark Zhandry. Unclonable decryption keys. *IACR Cryptol. ePrint Arch.*, 2020:877, 2020. (Cited on page 3.)

[HJL21] Samuel B. Hopkins, Aayush Jain, and Huijia Lin. Counterexamples to new circular security assumptions underlying iO. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 673–700, Virtual Event, August 2021. Springer, Heidelberg. (Cited on page 2.)

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012. (Cited on page 18.)

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 89–114. Springer, Heidelberg, August 2019. (Cited on page 5.)

[PW11] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. *SIAM Journal on Computing*, 40(6):1803–1844, 2011. (Cited on page 5.)

[RS19] Roy Radian and Or Sattath. Semi-quantum money. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies, AFT 2019*, pages 132–146. ACM, 2019. (Cited on page 2, 3, 9, 21.)

[RZ21] Bhaskar Roberts and Mark Zhandry. Franchised quantum money. *Asiacrypt 2021 (to appear)*, 2021. https://www.cs.princeton.edu/~mzhandry/docs/papers/Z21b.pdf. (Cited on page 7, 18.)

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 127–156. Springer, Heidelberg, October 2021. (Cited on page 2.)

[Zha21] Mark Zhandry. Quantum lightning never strikes the same state twice. or: Quantum money from cryptographic assumptions. *Journal of Cryptology*, 34(1):6, January 2021. (Cited on page 2, 3, 4, 6, 7, 17.)