Public-Key Quantum Fire and Key-Fire From Classical Oracles

Alper Çakan* Vipul Goyal† Omri Shmueli‡

Abstract

Quantum fire is a distribution of quantum states that can be efficiently cloned, but cannot be efficiently converted into a classical string. First considered by Nehoran and Zhandry (ITCS'24) and later formalized by Bostanci, Nehoran, Zhandry (STOC'25), quantum fire has strong applications and implications in cryptography, along with important connections to physics and complexity. However, constructing and proving the security of quantum fire so far has been elusive. Nehoran and Zhandry showed how to construct quantum fire relative to an *inefficient quantum oracle* (which cannot be instantiated even heuristically). Later, Bostanci, Nehoran, Zhandry gave a candidate construction based on group actions, however, even in the classical oracle model they could only conjecture the security of their scheme, and were not able to give a security proof or argue security.

In this work, for the first time, we give a construction of public-key quantum fire relative to a classical oracle and prove its security unconditionally. Going further, we introduce two stronger notions that generalize quantum fire, and we give secure constructions for these notions.

- We introduce a notion called *quantum key-fire* where the clonable fire states serve as *keys*: They can be used to evaluate a functionality (such as a signing or decryption key), and for security we require *unbounded leakage-resilience*, which means that given the fire state, the key cannot be efficiently converted into a classical string.
- We consider the notion of *interactive (i.e. LOCC) security* for quantum (key-)fire. In this setting, instead of trying to convert the state or key into a classical string; the adversary, given the flame state, attempts to transfer it to another adversary interactively over a classical channel.

We give a construction of quantum *key-fire* relative to a classical oracle and unconditionally prove that it satisfies interactive security for any unlearnable functionality. As a result, we also obtain the *first classical oracle separations* between various notions in physics and cryptography:

- A separation in the computational universe between two fundamental principles of quantum mechanics: No-cloning and no-teleportation, which are equivalent in the information-theoretic setting.
- A separation between *copy-protection security* (Aaronson, CCC'09) and *unbounded/LOCC leakage-resilience security* (Çakan, Goyal, Liu-Zhang, Ribeiro, TCC'24).
- A separation between *computational no-cloning security* and *computational no-learning security*, two notions introduced recently by Fefferman, Ghosh, Sinha, Yuen (ITCS'26).

In all our constructions, the oracles can be implemented efficiently using one-way functions. Thus, our schemes can be heuristically instantiated in the plain model using one-way functions and indistinguishability obfuscation, which gives us the first constructions in the plain model with concrete evidence for their security.

Keywords: Quantum cryptography, leakage-resilience, quantum protection

*Carnegie Mellon University. alpercakan98@gmail.com.
†NTT Research & Carnegie Mellon University. vipul@vipulgoyal.org
‡NTT Research. omri.shmueli1@gmail.com.

Contents

1 Introduction **4**
- 1.1 Our Results . . . 5
- 1.2 Applications, Implications and Future Directions . . . 8
- 1.3 Organization . . . 10

2 Technical Overview **11**
- 2.1 Definitions . . . 11
- 2.2 Construction . . . 12
- 2.3 Proving Security . . . 14
- 2.4 Interactive Security . . . 17
- 2.5 Quantum Key-Fire . . . 18
- 2.6 Related Work . . . 18

3 Preliminaries **20**
- 3.1 Notation and Conventions . . . 20
- 3.2 Pseudorandom Functions . . . 21
- 3.3 Digital Signature Schemes . . . 23
- 3.4 Indistinguishability Obfuscation . . . 23
- 3.5 Quantum Information . . . 24

I Definitions **26**

4 Quantum Protection Definitions **26**
- 4.1 Quantum Protection . . . 26
- 4.2 LOCC Leakage-Resilience . . . 27
- 4.3 One-Shot Signatures . . . 28

5 Definitional Work on Quantum Fire **30**
- 5.1 Quantum Fire . . . 30
- 5.2 Interactive Untelegraphability Security . . . 32
- 5.3 Quantum Key-Fire . . . 33
  - 5.3.1 Quantum Key-Fire for Signing . . . 34

II Technical Tools **36**

6 One-Shot Signatures with Incompressibility Security **36**
- 6.1 Operational Incompressibility . . . 37
- 6.2 Construction and Proofs of Security . . . 38
- 6.3 Properties of Subspace Oracles . . . 41
- 6.4 Proof of Incompressibility . . . 42

7 Incompressibility of Random Oracles **47**

III Quantum-Key Fire **52**

8 Quantum Key-Fire for Signing in the Classical Oracle Model **52**

9 Construction **54**
9.1 Building Block: Incompressible One-Shot Signature Scheme 54
9.2 Construction 55
9.3 Proof of Signing Correctness 59
9.4 Proof of Strong Cloning Correctness 59

10 Proof of Security of the Main Construction **64**
10.1 Defining Modified Adversaries 65
10.2 Defining the Hybrids 71
10.3 Hybrid Indistinguishability (Proof of Theorem 19) 76
10.3.1 Proof of Lemma 8 77
10.3.2 Proof of Lemma 9 79
10.3.3 Proof of Lemma 10 82
10.4 Proof of Theorem 20 84

11 Proof of Lemma 7 **86**
11.1 Proof of Lemma 15 88
11.2 Proof of Lemma 16 91

1 Introduction

Possibly the two most well-known basic principles of quantum mechanics are the so-called *no-cloning* and the *no-teleportation* principles. The first principle says that no physical process can create two copies of an unknown quantum state when given a single copy, and the second principle (which we will refer to as the *no-telegraphing principle*[1]) says that no physical process, given a single copy of a quantum state, can convert it into a classical string from which the state can be reconstructed.

Apart from being fundamental rules of the quantum universe we live in, these principles have also found a tremendous amount of applications in quantum cryptography. The no-cloning principle has had profound applications, from being utilized to construct *quantum money* in the very first paper of [Wie83] that started the field of quantum cryptography, to fundamental applications like quantum key-distribution and even more modern applications such as *copy-protection (anti-piracy security) for software* ([Aar09, CLLZ21, ÇG25]). The no-telegraphing principle has also found important applications in cryptography, such as the notion of *unbounded/LOCC leakage-resilience* ([ÇGLZR24, NZ24]).

It is well-known in quantum information theory that the no-cloning and the no-telegraphing principles are equivalent: If one can telegraph a state then they can also clone it, and if one can clone a state they can also telegraph it by performing state tomography after creating a sufficient number of copies of the state. One caveat is that state tomography necessarily requires exponentially many copies in the number of qubits ([FGLE12, OW16]), and the resulting classical string (describing the state) will also be of exponential size. Thus, the equivalency between the two principles relies on an inefficient physical process. However, it is one of the fundamental hypotheses of quantum complexity theory that any physical process we can observe in real-life is *quantum polynomial time*. Thus, one might ask the following question: Are these two principles of quantum mechanics equivalent through physically-observable processes (or, *reasonable-size experiments* in the terminology of [BBJ16])? Indeed, Nehoran and Zhandry ([NZ24], ITCS'24) precisely asked this question and showed that the answer is likely negative, by showing a separation relative to an inefficient quantum oracle. While this is a step towards answering the question, unfortunately, quantum oracle separations might be misleading ([AK25]) and in fact it is even unclear how exactly to model them ([Zha25]). Later, Bostanci, Nehoran, Zhandry ([BNZ25], STOC'25) gave a candidate classical oracle separation based on *cryptographic group actions*; however, they were not able to give a proof or argue security. Thus, what we know so far about the relationship between these two principles in a computational universe is unsatisfactory, and thus we ask the following question:

Can we show a separation between the no-cloning and the no-telegraphing principles in a computational universe?

Note that this question has fundamental implications for our understanding of quantum states, as well as to cryptography and complexity (as we explain in detail in the subsequent sections). It is widely believed that quantum states are more *powerful* than classical strings (e.g. the well-known QMA vs QCMA question). However, what exactly is the fundamental property of quantum states that makes them more powerful than classical strings? Is it unclonability? If we show a clonable state that satisfies the no-telegraphing principle, then the answer is negative: Even clonable quantum states are more powerful then classical strings.

[1] We follow this naming convention that [NZ24] introduced to avoid confusion with *quantum teleportation*

Cryptographic Applications. [BNZ25] defined a notion called *quantum fire*, which can be thought of a cryptographic version of the aforementioned separation[2]. In more detail, quantum fire is defined to be a distribution of quantum states (called flame states) which can be efficiently cloned. For security, we require *untelegraphability*: No efficient adversary who is given a flame state can convert it into a classical string from which the state can be reconstructed. [BNZ25] mention possible applications of quantum fire, such as storing some information in a quantum encoding (i.e. as a flame state) on a server to protect it from *exfiltration* by adversaries. However, the state of affairs for existence of quantum fire is unsatisfactory: Just like above, the only known constructions are the inefficient quantum oracle based construction of [NZ24] and the candidate construction of [BNZ25] without a proof of security. Thus, we ask the following question.

Can we construct quantum fire with provable security?

Given the myriad of applications of the no-cloning and the no-telegraphing notions in cryptography, we also ask whether a separation between them also has applications in cryptography, beyond quantum fire.

Does a separation between no-cloning and no-telegraphing have applications in cryptography?

1.1 Our Results

In this work, we answer the above questions in the classical oracle model.

Quantum Fire We give a construction of quantum fire relative to a quantumly-accessible[3] classical oracle and prove its correctness and security unconditionally, for the first time.

Theorem 1 (Informal)**.** *Relative to a classical oracle, there exists a public-key quantum fire scheme with untelegraphability security.*

This also gives the first classical oracle separation between no-cloning and no-telegraphing and resolves the open questions posed by [NZ24] and [BNZ25]. We note that our construction is completely different from previous works.

Our scheme can be implemented efficiently using one-way functions, and thus can be instantiated heuristically using indistinguishability obfuscation to obtain the first quantum fire scheme in the plain model with concrete evidence for its security.

This significantly improves upon existing results. As discussed above, previously [NZ24] gave a construction of quantum fire relative to an inefficient unitary quantum oracle (which bakes cloning into the oracle itself and thus it is inherently inefficient), and [BNZ25] gave a *candidate* construction based on cryptographic group actions. However, even in the classical oracle model, [BNZ25] only conjectured security and were not able to prove security. In fact, they even claimed that any quantum fire scheme with proof of security in the classical oracle model would likely lead to a classical oracle separation between QMA and QCMA[4], which is one of the most biggest questions in quantum complexity that has been open for almost 20 years[5].

[2] Indeed, quantum fire is stronger than the separation, as it immediately implies the separation.

[3] Throughout the paper, we will write classical oracle for brevity, and we always mean quantumly-accessible classical oracle.

[4] We note that our construction does not yield a separation between QMA and QCMA. However, it is an interesting question if our techniques could be useful.

[5] Subsequent to our work, a new work by Bostanci, Haferkamp, Nirkhe, Zhandry (STOC'26) solved this problem via a different approach.

Stronger Notions Going further, we introduce two stronger notions that generalize quantum fire.

- First, we introduce **quantum key-fire**, where the fire states encode a key that can be used to evaluate a functionality (e.g. a signing key, or a decryption key, or a software). For quantum key-fire, as the security notion, instead of untelegraphability, we require the stronger guarantee of **unbounded leakage-resilience** ([ÇGLZR24]). Unbounded leakage-resilience is a strengthening of no-telegraphing where not only that the flame state cannot be encoded into a classical string, we further require that no efficiently generated classical string (called *leakage* in this context) can help evaluate the functionality or win a security game related to the functionality. In other words, quantum key-fire is to quantum fire what quantum copy-protection ([Aar09, ALL+21, CLLZ21]) is to public-key quantum money, and quantum-key fire implies quantum fire in a straightforward way.

- As the second generalization, we consider the notion of **interactive security**. Previous work only considered the single message untelegraphability security game where the adversary $\mathcal{A}_1$ who has the flame state (the sender adversary) produces a classical string, using which the second adversary $\mathcal{A}_2$ (the receiver adversary) tries to reconstruct the flame state. We consider the case where the sender and the receiver interact over a classical channel for any (polynomial) number of rounds. We call this notion *interactive untelegraphability security* in the case of quantum fire, and *local operations and classical communication (LOCC) leakage-resilience* ([ÇGLZR24]) in the case of quantum key-fire.

To recap, we have four security notions for quantum fire and key-fire: **Untelegraphability, interactive untelegraphability, unbounded leakage-resilience and LOCC leakage-resilience** - a summary is given in Table 1. Note that LOCC leakage-resilience implies interactive untelegraphability, and unbounded leakage-resilience implies untelegraphability.

Security Notion	**Interaction Between $\mathcal{A}_1$ and $\mathcal{A}_2$**	**Goal of $\mathcal{A}_2$**
Untelegraphability	No	Reconstruct the exact same state
Interactive Untelegraphability	Arbitrary classical interaction	Reconstruct the exact same state
Unbounded Leakage Resilience	No	Solve a challenge (e.g. decrypt a challenge ciphertext)
LOCC Leakage Resilience (the strongest notion)	Arbitrary classical interaction	Solve a challenge (e.g. decrypt a challenge ciphertext)

Table 1: Comparison of security notions for quantum fire and key-fire.

Quantum Key-Fire We give a construction of quantum key-fire relative to a classical oracle and prove that it satisfies LOCC leakage-resilience for all unlearnable functionalities.

Theorem 2 (Informal)**.** *Relative to a classical oracle, there exists a quantum key-fire scheme that satisfies LOCC leakage-resilience for any unlearnable functionality.*

We note that this is the best possible result since even in the oracle model, unbounded/LOCC leakage-resilience for learnable functionalities is trivially impossible.

As above, our scheme can be instantiated efficiently using one-way functions and indistinguishability obfuscation in the plain model with heuristic security.

Previously, even in the quantum oracle model, only provably secure construction of quantum fire [NZ24] had proof of security in the significantly weaker setting of single-message untelegraphability and there was no proof of even 2-message interactive security.

Size of the Leakage Finally, we note that, unlike previous work, we impose no restriction on the length of the messages between the adversaries (e.g. they can be doubly exponential in the security parameter, or more): The only requirement for security is that the adversaries make polynomially many queries to the oracles.

Applications As we will discuss later in more detail (see Section 1.2), quantum key-fire is a strong primitive in the setting of *leakage-resilient cryptography*. It retains the security of the functionality (e.g. a decryption key) even in the face of *unbounded* leakages, which is impossible with classical keys, but at the same time (assuming one has a quantum computer) it is almost as practical as a classical key due to its clonability: It can be later on used in different locations/servers, can be kept alive for long periods by frequently refreshing/cloning it (almost like a classical dynamic RAM) and so on.

Implications As we discuss in Section 1.2, our results have interesting implications in cryptography, physics, and complexity, and we believe it also leads the way for a myriad of interesting new research directions (such as *clonable quantum cryptography*) and questions.

As an example of cryptographic implications, our results imply a separation between the recently introduced cryptographic assumptions of *computational no-cloning assumption* and *computational no-learning assumption*, thus answering an open question of [FGSY25]. Our results also imply a separation between the security notions of *copy-protection security* and *unbounded/LOCC leakage-resilience security*, answering the open questions of [NZ24] and [ÇGLZR24]. As an example of physical implications, our result shows that the fundamental property that makes quantum states more *powerful* than classical strings is not unclonability (since even clonable states cannot be converted into classical strings).

Technical Contributions Our work introduces new techniques in the oracle model for analyzing two quantum adversaries that are communicating interactively for arbitrary number of rounds over a classical channel (i.e. LOCC adversaries), which are in general hard to analyze due to internal unclonable states of the adversaries. Through our new techniques, we also draw interesting connections between LOCC adversaries and a natural property called *incompressibility* of oracles that says that not *too many outputs* of the oracle can be packed inside a classical string (see Sections 6 and 7 for the formal definitions). We also prove that our incompressibility notion is satisfied for two particular cases: for random oracles and for a modified version of the one-shot signature scheme of [SZ25].

We believe our new techniques and our new incompressibility results might be helpful for other problems in similar settings.

Theorem 3 (Informal)**.** *Random oracles satisfy classical-message incompressibility.*

Theorem 4 (Informal)**.** *Relative to a classical oracle, there exists a one-shot signature scheme with incompressibility.*

1.2 Applications, Implications and Future Directions

In this section, we will discuss the cryptographic applications (e.g. constructions) of our results, and then discuss the implications of our results (e.g. separations we obtain) and finally discuss some new research directions motivated by our results.

We first recall the notion of leakage-resilience.

Primer on Leakage-Resilience Storing secret keys (e.g. of an encryption scheme or a signature scheme) is one of the oldest and most fundamental problems in cryptography. Leakage-resilient cryptography is a fruitful research area in cryptography that has received significant interest over decades that aims to deal with this problem: It aims to construct schemes which are secure in face of attacks where adversary obtains *leakage* on the honest parties secret keys, through *side-channels* [Koc96, QS01, AARR03]. Side-channels are generally various measurements of the system storing the keys, such as the electromagnetic radiation emitted, or the power consumed during computation.

In classical leakage-resilient cryptography, the leakage output is modeled as as function $f(sk)$ of the secret key sk where f has an output size significantly smaller than the size of sk. This is a somewhat artificial restriction that is nevertheless ubiquitous, since otherwise there is no possibility of obtaining security guarantees: the adversary can simply obtain the whole secret key.

The recent work of [ÇGLZR24] has introduced[6] the notion of *unbounded leakage-resilience*, where the secret key is stored as a quantum key, and the leakage measurement output is allowed to be of any unbounded size[7], in particular, it can be much larger than the size of the secret key! They also introduced the notion of *LOCC leakage-resilience* where the adversary adaptively obtains unbounded leakage over any number of rounds. They showed, through a new proof, that the unclonable[8] public-key encryption and signing schemes of [CLLZ21] also satisfy LOCC leakage-resilience, thus giving a plain model construction assuming indistinguishability obfuscation (iO) and one-way functions.

Application: Unbounded/LOCC Leakage-Resilience with Clonable Keys The notion of *unbounded/LOCC leakage-resilience* is a strong (and uniquely quantum) security guarantee with important practical ramifications. However, a major issue with the existing scheme of [ÇGLZR24] is that while the secret quantum key satisfies unbounded/LOCC leakage resilience, the key is also unclonable. This severely limits the practicality of the scheme, since if we ever need to use the decryption key in multiple systems, this will not be possible[9]. Further, while there have been significant advances in quantum computation in recent years, quantum memory implementations can still keep states coherent for very short amounts of time.

However, a quantum key-fire scheme gives us the best of both worlds: While the keys satisfy LOCC leakage-resilience (which is impossible classically), they are also efficiently clonable, which makes it almost as useful as a classical key! For example, the key can be used in different computers too (e.g. to parallelize big tasks, or if these are the decryption keys for a company and a new employee joins). Further, they can be frequently cloned in the quantum memory to *refresh* them to keep them alive for much longer and prevent decoherence, much like a classical *dynamic RAM*

[6] Concurrent to [ÇGLZR24], [NZ24] introduced *exfiltration security*, which is essentially equivalent to non-interactive unbounded leakage-resilience. They did not give a plain model construction from standard cryptographic assumptions.

[7] The only requirements is that leakage is a measurement, i.e., it has a classical output. However, measurements accurately model side-channel attacks and in fact capture all known attacks.

[8] That is, the decryption/signing keys of the schemes satisfy *copy-protection security*

[9] Naturally, we cannot store a classical description of the key either, since that would defeat the whole purpose and we would lose the unbounded leakage-resilience guarantee.

that are widely used in practice. Thus, we believe quantum key-fire will have important practical ramifications in the future.

Separating Leakage-Resilience and Copy-Protection Related to above discussion, our quantum key-fire result also gives a separation between the security notions of unbounded/LOCC leakage-resilience ([ÇGLZR24]) and copy-protection ([Aar09]), answering an open question of [ÇGLZR24, NZ24]. Note that so far, only known unbounded/LOCC leakage-resilient schemes also satisfy copy-protection security. Copy-protected schemes are known to imply public-key quantum money (PKQM), and constructing PKQM without obfuscation is one of the biggest problems in quantum cryptography. However, our work for the first time shows that leakage-resilience does not imply copy-protection/unclonability, which means there might be a hope of achieving unbounded/LOCC leakage-resilience without obfuscation. On the (slightly) negative side, our separation shows that we cannot simply prove unbounded/LOCC leakage-resilience of a scheme and be content that it also satisfies copy-protection security: when the goal is to obtain copy-protection security, we need to prove it separately.

Clonable Cryptography and Separations in MicroCrypt MicroCrypt is the study of existence of quantum cryptographic primitives (such as quantum commitments, one-way quantum-state generators etc) even in a world where $\mathsf{P} = \mathsf{NP}$, and it has gained significant attention in the recent years. Recent work of [FGSY25] introduced two new quantum cryptographic assumptions for MicroCrypt: the computational no-learning assumption and the computational no-cloning assumption, and they show how to build various MicroCrypt primitives from these assumptions. They also conjecture that the computational no-learning assumption is potentially weaker (i.e., it is a more conservative assumption). Our work shows that relative to a classical oracle, there exists a family of circuits for which the computational no-learning holds, but the computational no-cloning assumption does not. This resolves their open question. We believe our techniques might also help separate various MicroCrypt primitives, such as one-way state generators and unclonable state generators.

We believe our work will also lead the way for a new research direction: *clonable quantum cryptography*. Our work shows that even clonable states are more powerful than classical strings. Thus, **in any context where the purpose of using quantum states is not unclonability itself, it might be possible to obtain a clonable version of that quantum primitive, which would make it more practical** (for example, the state can be kept alive for longer by refreshing it through cloning it, much like a classical dynamic RAM). For example, there are various works [GMMY24, MY22, KMNY24] that show that various primitives (such as public-key encryption, signatures) can exist with quantum keys or quantum ciphertexts/signatures even if their classical-string counterparts do not exist. However, all such candidates so far have unclonable keys/signatures/ciphertexts, which significantly limits their practicality. Thus, it is an interesting question if one can show existence of clonable versions of these primitives in a world their classical counterpart do not exist, and we believe our techniques might help.

Physics in a Computational Universe In quantum mechanics, various fundamental principles called *no-go theorems* exist, such as the *no-cloning theorem*, *no-hiding theorem*, *no-deleting theorem*, *no-telegraphing theorem* and so on. Most of these no-go theorems have been shown to be equivalent to each other in the information-theoretic setting. However, our result gives the first (classical oracle) separation between the *no-go principles* of quantum mechanics. Our result means that, relative to a classical oracle, no-telegraphing does not imply no-cloning. Our result means that the

equivalence of the fundamental no-go principles of quantum mechanics that we take for granted might not hold in a computational universe. An interesting question is if other principles of quantum mechanics can be separated in a computational universe.

Another interpretation of our result is as follows. In an information-theoretic setting, the main difference between a (local) quantum state/memory and a classical string is the unclonability of the quantum state, and clonable states can simply be turned into classical strings: We can clone it and apply a state tomography to obtain a classical description of it, and then work with this classical string. However, our work shows that in a computational world, even clonable states are different from classical memory, thus the fundamental property that distinguishes quantum states from classical memory is not unclonability.

Complexity Separating the complexity classes QMA and QCMA relative to a classical oracle is one of the most important questions in quantum cryptography and has been open since 2007. Due to the connection established by [NZ24] between quantum fire and separating clonableQMA from QCMA (which also separates QMA and QCMA), [BNZ25] claimed that a quantum fire scheme with proof of security in the classical oracle model would likely lead to a classical oracle separation between QMA and QCMA, solving this major open question. While our construction does not lead to such a separation, it is an interesting question if our techniques could help[10].

1.3 Organization

In Section 2, we give a technical overview of the paper.

In Section 4, we recall some prior definitions, such as quantum protection schemes and LOCC leakage-resilience security, that are relevant to defining and constructing quantum fire. In Section 5, we give the quantum fire and quantum key-fire definitions.

In Section 7, we prove an incompressibility result for random oracles. In Section 6, we define and construct incompressible one-shot signatures.

In Section 9, we give our main construction: quantum key-fire for signing. Then, we prove its verification and cloning correctness.

In Section 10, we prove the security of our main construction, except for one major hybrid, which we prove in Section 11 for ease of reading.

Acknowledgments

AÇ thanks Barak Nehoran for answering questions about their work ([BNZ25]), Christian Majenz for the correct citation for the gentle measurement lemma ([Win99]) and Dakshita Khurana for helpful discussions. We thank Vinod Vaikuntanathan and Andrew Huang for helpful comments.

AÇ was supported by the following grants of Vipul Goyal: NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award. This work was done in part while OS was a research fellow at the Simons Institute for the Theory of Computing at UC Berkeley.

[10]Subsequent to initial posting of our work, Bostanci, Haferkamp, Nirkhe, Zhandry (STOC'26) solved this problem via a different approach, though the core philosophical principle under their proof is similar to ours: Classical leakage/witnesses are by default clonable for free and reusable, so we can run an adversary many times with it, unlike quantum states.

2 Technical Overview

In this section, we give an overview of our work, and the end of the of the section we discuss some related work. Formal definitions, constructions and proofs can be found in relevant sections (see Section 1.3 for the organization). Note that for most of the overview, for pedagogical reasons, we will consider the notion of quantum fire mini-schemes with untelegraphability security. However, in the main body of the paper, we instead construct quantum key-fire and prove LOCC leakage-resilience security, which is stronger.

Notation We follow the common math/cryptography notation - more details can be found in Section 3.1, but importantly, we write R to denote quantum registers, and variables written in lower-case letters will be classical objects, such as sk, pk. Further, *efficient* adversary means quantum polynomial time (QPT) in the plain model and in the oracle model it means an adversary who makes polynomially many quantum queries to the oracles (it can take unbounded time outside oracle calls). We write $\mathcal{O}$ to denote a classical oracle and write $\mathcal{A}^{\mathcal{O}}$ or $\mathcal{B}^{\mathcal{O}}$ to denote an adversary or algorithm with quantum query access to $\mathcal{O}$. Throughout the paper, oracle will always mean quantumly-accessible classical oracle (even when do not explicitly write so).

2.1 Definitions

We first start by providing an overview of the primitives and security definitions we consider in this overview - the formal definitions can be found in Section 5. The definition of one-shot signatures (which we use as a building block) can be found in Definition 9.

Quantum Fire Mini-Scheme In the first parts of this overview, we will work with a notion called *quantum fire mini-scheme* (Definition 14), which consists of three efficient quantum algorithms: Spark, Ver, Clone. $\mathsf{QFireMini.Spark}(1^\lambda)$ outputs a serial number sn^* and a flame state $|\psi_{sn^*}\rangle$. $\mathsf{QFire.Ver}(sn, \mathsf{R})$ verifies if the state in the register R is a valid flame state for the serial number sn. Finally, $\mathsf{QFire.Clone}(sn, \mathsf{R}_{\mathsf{flame}})$ outputs a register $\mathsf{R}_{\mathsf{clone}}$, which contains a copy of the the flame state in the register $\mathsf{R}_{\mathsf{flame}}$, along with outputting the register $\mathsf{R}_{\mathsf{flame}}$ back. We require verification correctness and cloning correctness. Verification correctness requires that honestly generated states pass verification of Ver with overwhelming probability. For cloning correctness, we require that the $\mathsf{Clone}(sn, |\psi_{sn}\rangle)$ outputs $|\psi_{sn}\rangle \otimes |\psi_{sn}\rangle$.

Untelegraphability Security In this overview, we will consider *untelegraphability security* (Definitions 13 and 15), which is defined by the following security game between a challenger and a pair of efficient adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. The sender adversary $\mathcal{A}_1$ is given $sn^*, |\psi_{sn^*}\rangle \leftarrow \mathsf{QFireMini.Spark}(1^\lambda)$ and produces a classical output L. Afterwards, the receiver adversary $\mathcal{A}_2(L)$ simply outputs a register R', which is tested for sn^* by running $\mathsf{QFireMini.Ver}(sn^*, \mathsf{R}')$. A scheme is said to satisfy untelegraphability security if for any efficient adversary $\mathcal{A}$, the probability of winning the game is negligibly small.

Full Schemes In a full fledged quantum fire scheme QFire (Definition 11), there is a secret key sk with which $\mathsf{QFire.Spark}$ is run. In the security game (Definition 13), the adversary $\mathcal{A}_1$ receives multiple independent flame states, and $\mathcal{A}_2$ tries to produce a flame state with *any* serial number (not necessarily one received by $\mathcal{A}_1$). Working with mini-schemes simplifies our discussion, and any mini-scheme can be generically upgraded to a secure full-fledged quantum fire scheme by using

a digital signature on the serial number sn: $\mathsf{QFire.Spark}(sk)$ outputs $\mathsf{Sign}(sk, sn)||sn, \mathsf{R}_{\mathsf{flame}}$ where $sn, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QFireMini.Spark}(1^\lambda)$.

2.2 Construction

We now give an overview of our quantum fire mini-scheme construction, slowly building it to highlight the main ideas and design choices. The full version of our main construction (a *key-fire* scheme for signing) be found in Section 9.

Intuitively, the notions of clonability and untelegraphability are at odds. The parties need to be able create new copies of the flame state $|\psi_{sn}\rangle$, however, since we have classical oracles, we cannot simply output new copies of the state directly, or output any quantum states for that matter. Since we need to satisfy untelegraphability, we cannot output a description of the state directly either. Indeed, it is overall unclear how a classical oracle can help synthesize a state without simply outputting the description of the state in some form.

Starting Point As a starting point, we turn our attention to *one-shot signature* (OSS) schemes (Definitions 9 and 10), which has been recently constructed in the classical oracle model by [SZ25]. In a one-shot signature scheme, using a classical oracle $\mathcal{O}$, parties are able to *sample* a single copy of a quantum state (called signing keys) $|\phi_{vk}\rangle$ for some random verification key vk, by running the algorithm $\mathsf{OSS}^{\mathcal{O}}.\mathsf{GenQKey}(1^\lambda)$. Then, they can sign any message $m \in \{0,1\}$ using $|\phi_{vk}\rangle$, but the security requirement is that they cannot sign both 0 and 1 with the same vk value. In particular, this requires that $|\phi_{vk}\rangle$ cannot be cloned. Using OSS for quantum fire might seem counterintuitive: After all, the states $|\phi_{vk}\rangle$ are unclonable, which is the opposite of clonability(!), and a main application of OSS is classically-transferable quantum primitives (such as quantum money that can be transferred over classical channels), whereas untelegraphability is the polar opposite of this!

Indeed, while the focus of OSS schemes is unclonability of the states $|\phi_{vk}\rangle$, we take a new viewpoint: We observe that the OSS security also guarantees that parties can sample $|\phi_{vk}\rangle$ without learning its classical description! Thus, we consider using a one-shot signature signing key $|\phi_{vk}\rangle$ as our flame states, and we hope to use $\mathsf{GenQKey}$ as our cloning algorithm. However, there are two major issues with this approach. First, the state generation algorithm $\mathsf{GenQKey}$ outputs a different state $|\phi_{vk'}\rangle$ each time (necessarily, since otherwise OSS security would be broken) rather than outputting a copy of our state! While we solve this problem by using the *purified* output $|\eta\rangle$ of $\mathsf{OSS}^{\mathcal{O}}.\mathsf{GenQKey}(1^\lambda)$ (which is essentially a superposition $\sum_{vk} |vk\rangle |\phi_{vk}\rangle$ of all possible keys), the second issue is much more significant: $\mathsf{OSS.GenQKey}$ is a public algorithm, and thus anyone can create $|\eta\rangle$! Thus, the sender adversary $\mathcal{A}_1^{\mathcal{O}}$ for quantum fire will not even need to prepare a classical encoding, since $\mathcal{A}_2^{\mathcal{O}}$ can simply recreate the public state $|\eta\rangle$ on its own, even though neither actually knows a classical encoding of it. To solve this issue, one might consider defining a secret key version of one-shot signatures (e.g. $\mathsf{OSS.GenQKey}$ requires sk), however, this would be of no help: (i) the secret key itself would serve as a classical encoding from which $\mathcal{A}_2$ can recreate $|\eta\rangle$, and more importantly (ii) the flame state must be publicly clonable, thus, we would need a new way to generate states without revealing the secret key, taking us back to square one.

Self-Encrypted Oracles To overcome these issues, we consider *encrypting* the oracle $\mathcal{O}_{\mathsf{OSS}}$ of OSS into a new oracle $\mathcal{O}'$ in a way so that the oracle can be evaluated only if one *proves* that they already have the flame state $|\eta\rangle$ (we will call this a $|\eta\rangle$-verification mechanism). With this approach, we would at least eliminate the trivial attack where the receiver adversary $\mathcal{A}_2$ can (without any message from $\mathcal{A}_1$) create $|\eta\rangle$ on their own. Further, (publicly) cloning would still be possible, since

(hopefully) one would be able to *simulate* a run of $\mathsf{OSS.GenQKey}^{\mathcal{O}_{\mathsf{OSS}}}(1^\lambda)$ by decrypting outputs of $\mathcal{O}'$ using their initial state $|\eta\rangle$ and passing them along to $\mathsf{OSS.GenQKey}$.

However, there are still major challenges with this approach. First, it is not immediately clear how one can *prove* to a *classical* oracle that they posses a quantum state. Second, even if one can achieve a *sound* oracle encryption, the mechanism used to achieve this and implement the encrypted oracle $\mathcal{O}'$ must be so that we can actually decrypt and evaluate the original oracle $\mathcal{O}_{\mathsf{OSS}}$ *correctly* even for *quantum queries*, since $\mathsf{OSS.GenQKey}$ will have quantum queries to the oracle $\mathcal{O}_{\mathsf{OSS}}$. Further, the $|\eta\rangle$-verification mechanism of our encrypted oracle will need to use (in some form) the verification mechanism of OSS itself (such as $\mathsf{OSS.Ver}$), however, these algorithms also need access to $\mathcal{O}_{\mathsf{OSS}}$, thus we will need to leave some parts of the oracle unencrypted without revealing too much information. Third, once we try to prove security, we run into a major circularity issue: While we want to say that the receiver adversary $\mathcal{A}_2$ will not be able to create the state $|\eta\rangle$ since it cannot decrypt the oracle $\mathcal{O}'$; to show that $\mathcal{A}_2$ cannot decrypt $\mathcal{O}'$, (since this oracle is encrypted under $|\eta\rangle$) we need to show that it cannot have $|\eta\rangle$ to begin with!

Our Oracles We now give an overview of our oracle encryption mechanism that solves all of these issues. We first note that our oracle encryption mechanism will not fully hide the oracle $\mathcal{O}_{\mathsf{OSS}}$ from $\mathcal{A}_2$: It will be able to evaluate it, but we will show that it can only do so at few points, which will be sufficient to prove that it cannot produce $|\eta\rangle$,

Now we move onto our oracles. First, we require that the OSS scheme's oracle $\mathcal{O}_{\mathsf{OSS}}$ consists of three parts: $\mathcal{O}_{\mathsf{OSS.Ver}}, \mathcal{O}_{\mathsf{OSS.Sign}}$ and $\mathcal{O}_{\mathsf{OSS.GenQKey}}$ where each of the three algorithms ($\mathsf{OSS.Ver}$, $\mathsf{OSS.Sign}$, $\mathsf{OSS.GenQKey}$) uses only the corresponding oracle. Of course, setting all of these oracles to be the full oracle of the scheme would also satisfy this, but as discussed, we require a non-trivial splitting of the oracle so that (i) $\mathcal{O}_{\mathsf{OSS.Ver}}, \mathcal{O}_{\mathsf{OSS.Sign}}$ do not reveal too much information (more details when we discuss the proof) and (ii) $\mathcal{O}_{\mathsf{OSS.Ver}}, \mathcal{O}_{\mathsf{OSS.Sign}}$ are sufficient for our $|\eta\rangle$-verification mechanism. Then, we define the following three oracles $\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2$ as our encrypted oracle $\mathcal{O}'$. On input $vk, isig, z$, $\mathcal{O}_b$ (for $b \in \{0,1\}$) verifies (using $\mathcal{O}_{\mathsf{OSS.Ver}}$) that $isig$ is as valid OSS signature for the message b with verification key vk, and then it outputs $H_b(vk||z)$ where H_0, H_1 are random functions. We will call the values $H_b(vk||z)$ *attestation strings*. Finally, on input vk, y_0, y_1, z, $\mathcal{O}_2$ verifies that $y_0 = H_0(vk||z)$, $y_1 = H_1(vk||z)$, and then outputs $\mathcal{O}_{\mathsf{OSS.GenQKey}}(z)$.

Essentially, our encrypted oracle verifies that the party has a signature for both 0 and 1 (by verifying the corresponding attestation strings) with the same verification key, and then releases the output of the original oracle $\mathcal{O}_{\mathsf{OSS.GenQKey}}(z)$. Later, we will argue that this is indeed a sound $|\eta\rangle$-verification mechanism.

Design Choices: What Does Not Work Here, we motivate our oracle design given above by mentioning some other possible choices that do not work. First, note that using two separate oracles $\mathcal{O}_0, \mathcal{O}_1$ is required: We cannot have a single oracle that asks for a signature of 0 and 1 with the same verification key, since by the security of the one-shot signature scheme, we would not be able to unlock such an oracle encryption. Also note that it would not be sound to accept a signature of 0 and 1 with different verification keys because it leads to a simple attack: After cloning $|\eta\rangle$ to get $|\eta\rangle \otimes |\eta\rangle$, the adversary $\mathcal{A}_1$ can measure the verification key registers to obtain $|\phi_{vk_1}\rangle, |\phi_{vk_2}\rangle$, and sign 0 and 1 with these keys respectively, and send the classical signatures to the receiver adversary $\mathcal{A}_2$. If we were to accept different verification keys in $\mathcal{O}_2$, the adversary $\mathcal{A}_2$ would be able to run the oracle $\mathcal{O}_2$ on any input z!

Cloning Now, we discuss cloning. To clone our flame state $|\eta\rangle = \sum_{vk} |vk\rangle \otimes |\phi_{vk}\rangle$, we simulate $\mathsf{OSS.GenQKey}$ while simulating its oracle queries using $\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2$. Suppose $\mathsf{OSS.GenQKey}$ makes the quantum query $|\tau\rangle = \sum \alpha_z |z\rangle$, which would normally be executed with $\mathcal{O}_{\mathsf{GenQKey}}$, but we simulate it as follows. We first run (a purified version) of $\mathsf{OSS.Sign}$ on $|\eta\rangle$ to convert it to a superposition $|\eta_0\rangle = \sum_{vk} |vk\rangle \otimes \left(\sum_{sig \in \mathsf{SIG}(vk,0)} |sig\rangle\right)$ of signatures for the message 0. Then, we query $\mathcal{O}_0$ on this state, which converts it to $|\eta_0'\rangle = \sum_{z,vk} \alpha_z |z\rangle |vk\rangle \otimes \left(\sum_{sig \in \mathsf{SIG}(vk,0)} |sig\rangle_{\mathsf{sig}} \otimes |H_0(vk||z)\rangle_{\mathsf{out_0}}\right)$ by definition of $\mathcal{O}_0$. Note that this can be also written as $|\eta_0'\rangle = \sum_{z,vk} \alpha_z |z\rangle |vk\rangle \otimes \left(\sum_{sig \in \mathsf{SIG}(vk,0)} |sig\rangle\right) \otimes |H_0(vk||z)\rangle_{\mathsf{out_0}}$ since the attestation string $H_0(vk||z)$ in the register $\mathsf{out_0}$ does not depend on the particular signature sig on the signature register sig. Now, since we have isolated output the attestation string, we can actually undo (by applying the inverse unitary) the signing operation to get our original key back $|\eta\rangle$! Then, we can similarly *sign* 1 and query $\mathcal{O}_1$ to get the second attestation and end up with $|\eta''\rangle = \sum_{z,vk} \alpha_z |z\rangle |vk\rangle \otimes \left(\sum_{sig \in \mathsf{SIG}(vk,1)} |sig\rangle\right) \otimes |H_0(vk||z)\rangle_{\mathsf{out_0}} \otimes |H_1(vk||z)\rangle_{\mathsf{out_1}}$. Finally, we can query $\mathcal{O}_2$ with both our attestations in place to get output of the actual oracle $\mathcal{O}_{\mathsf{GenQKey}}$. Afterwards, we show that we can again undo the signature to get our original key back, and be ready for decryption of the next query of $\mathcal{O}_{\mathsf{GenQKey}}$.

More Design Choices We note that above, for everything to work *smoothly*, it is crucial that (i) our attestation strings do not depend on the particular signature (i.e. it is $H_0(vk||z)$ and not $H_0(vk||isig||z)$) and (ii) we simply sign 0 and 1 rather than signing z (which would create an entanglement between the oracle input/output and our flame state because of the quantum queries of $\mathsf{GenQKey}$, so we would not be able move onto the second query with a *clean* flame state). Here, we only (informally) discussed how to handle the first query - the full proof of cloning correctness requires carefully formalizing the above arguments and is given in Section 9.4. We also note that, due to the design choice above necessitated by cloning correctness, as mentioned before our oracle encryption $\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2$ does not actually fully hide $\mathcal{O}_{\mathsf{OSS.GenQKey}}$ from $\mathcal{A}_2$. For example, $\mathcal{A}_1$ can actually obtain (for some vk) $H_0(vk||z), H_1(vk||z)$ for a list of z values, and send these to $\mathcal{A}_2$ with which $\mathcal{A}_2$ can obtain $\mathcal{O}_{\mathsf{OSS.GenQKey}}(z)$. However, we will show that this can happen only for a few points, and we will still be able to prove security.

Construction Summary To summarize, the oracles of our quantum fire construction $\mathsf{QFireMini}$ are $\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_{\mathsf{OSS.Ver}}, \mathcal{O}_{\mathsf{OSS.Sign}}$, and the flame state is the purified output $|\eta\rangle$ of $\mathsf{OSS.GenQKey}(1^\lambda)$. To verify an alleged flame state, $\mathsf{QFireMini.Ver}$ tries signing (using $\mathsf{OSS.Sign}$) first the message 0, and then rewinds and tries signing the message 1, and accepts if both succeed. We note that $\mathsf{QFireMini.Ver}$ does not only accept $|\eta\rangle$ - it actually accepts any OSS signing key $|\psi_{vk}\rangle$, but we prove that this is not a security issue. Finally, $\mathsf{QFireMini.Clone}$ proceeds by simulating $\mathsf{OSS.GenQKey}$, where oracle queries are simulated by forwarding them to $(\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2)$ and decrypting as described above. Note that we do not actually give a direct construction for $\mathsf{QFireMini}$, and the scheme summarized above was for exposition. Our full scheme is a key-fire scheme for signing and is given in Section 9.

2.3 Proving Security

We now give a high-level overview of our proof strategy: Our full security proof (Section 10) is highly delicate and technical. Note that our formal proof is for the stronger notion of quantum key-fire, whereas in this section for ease of exposition we will discuss the case of quantum fire mini-schemes.

As argued before, to prove security, we hope to show that [Step 1] $\mathcal{O}_2$ has a sound verification mechanism so that the receiver adversary $\mathcal{A}_2$ will not be able to use the oracle $\mathcal{O}_{\mathsf{OSS.GenQKey}}$ (except at few points) and [Step 2] the unencrypted parts of the original oracle ($\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}$) are not leaking *too much* information, that is, $\mathcal{A}_2$ is not able to produce $|\eta\rangle$ when it cannot unlock oracle $\mathcal{O}_{\mathsf{OSS.GenQKey}}$ (but can access $\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}$).

We first start with the second point. To isolate [Step 2], assume that the receiver adversary $\mathcal{A}_2$ has access to the oracles $\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}$, but does not have access to $\mathcal{O}_2$ (so, no access to $\mathcal{O}_{\mathsf{OSS.GenQKey}}$). Note that $\mathcal{A}_1$ still has access to all oracles ($\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}$) of quantum fire. In this case, it is still not immediately clear if $\mathcal{A}_2$ is not able to produce $|\eta\rangle$. After all, $\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}$ might be leaking too much information - indeed, the trivial splitting of the oracles of OSS would set them to be the same as the full oracle $\mathcal{O}_{\mathsf{OSS.GenQKey}}$. However, we prove that there is actually an OSS construction such that any $\mathcal{A}_2$ that has access to only $\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}$ and receives a classical message from $\mathcal{A}_1$ provably cannot produce a signing key $|\phi_{vk}\rangle$!

Incompressibility We note that the above security requirement is somewhat unnatural for an OSS scheme, since for one-shot signature schemes, there is no concept of secrecy (indeed, $|\eta\rangle$ is a publicly generatable state!). Thus, instead of proving the above directly, we introduce a stronger yet more natural property for OSS schemes called *incompressibility*.

Definition 1 (Incompressible One-Shot Signatures (Definitions 21 and 22, informal))**.** *A one-shot signature scheme relative to an oracle* $(\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}, \mathcal{O}_{\mathsf{OSS.GenQKey}})$ *is said to be* incompressible *if it satisfies the following for any efficient adversary* $\mathcal{A}_1, \mathcal{A}_2$*.*

$\mathcal{A}_1$ *gets access to* $\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}, \mathcal{O}_{\mathsf{OSS.GenQKey}}$*, and produces a classical output* L*. Then,* $\mathcal{A}_2$ *receives* L *and oracle access to only* $\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}$*. We require that there is a small (i.e. polynomial size) list of verification key values such that with overwhelming probability, any valid signature-verification key pair produced by any* $\mathcal{A}_2$ *on input* L must *come from* $\mathfrak{L}_L$*.*

In a sense, incompressibility says that not too many signatures can be packed in to a classical leakage string. Note that a quantum message analogue of this incompressibility notion is not achievable: $\mathcal{A}_1$ can simply send an equal superposition of exponentially many signatures.

We prove that the OSS construction of [SZ25], with a careful splitting of the oracles, satisfies incompressibility. For technical reasons that will be clear later on, we also prove an incompressibility property for random oracles (Section 7). We believe the notion of incompressibility and our results might be of independent interest. Our formal definitions, constructions and proofs are given in Section 7 and Section 6. Before moving on, we note that while incompressibility is a natural property, it is not automatically satisfied by any OSS scheme and thus requires a completely new proof separate from one-shot security. For example, one can think of an OSS scheme where the signatures and verification keys are rerandomizable, or an OSS scheme where $\mathcal{O}_{\mathsf{OSS.Sign}}$ additionally outputs a signature for a random verification key. None of these properties would cause a violation of one-shot signature security, but they would trivially break incompressibility.

Solving [Step 2] Observe that with the incompressibility property, we complete [Step 2] as follows. Suppose for a contradiction that $\mathcal{A}_2$ (with no access to $\mathcal{O}_2$) receives L from $\mathcal{A}_1$ and produces a state ρ that passes $\mathsf{QFireMini.Ver}$ with non-negligible probability. However, we can then run $\mathcal{A}_2(L)$ twice to obtain to two copies of ρ, and with one of them we can sign 0 to get $vk_1, isig_1$ and with the other one we can sign 1 to obtain $vk_2, isig_2$. Since we said $\mathsf{QFireMini.Ver}$ accepts ρ with non-negligible probability, we can then conclude that $(vk_1, isig_1)$ and $(vk_2, isig_2)$ will both be valid

with non-negligible probability by Jensen's inequality. However, by incompressibility property, we also know that vk_1 and vk_2 come from a small set, thus we can show that with non-negligible probability, vk_1 and vk_2 will be the same! This gives a contradiction to one-shot security, thus completing proof of [Step 2].

Oracle Encryption Security: Proving [Step 1] We essentially argued (in [Step 2]) that untelegraphability security is satisfied if $\mathcal{A}_2$ did not have access to $\mathcal{O}_{\mathsf{OSS.GenQKey}}$. Now, we consider [Step 1]: actually showing that our oracle encryption encryption ensures that this is the case. As discussed before, to do this, we want to rely on the fact that $\mathcal{O}_2$ in some sense verifies $|\eta\rangle$ before producing its output. However, we are trying to show that $\mathcal{A}_2$ cannot produce $|\eta\rangle$ to begin with! This circularity problem also persists when we try to utilize the incompressibility property: It only applies if $\mathcal{A}_2$ does not have access to $\mathcal{O}_{\mathsf{OSS.GenQKey}}$, but that's what we are trying to show to begin with!

To solve this issue and eliminate the circularity problem, we take a fine-grained approach: We consider the execution of $\mathcal{A}_2$ query-by-query. In particular, we consider[11] the queries of $\mathcal{A}_2$ to $\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2$ to proceed in triplets (with queries to $\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}$ in between the triplets). We will slowly remove these query triplets and instead simulate them with messages from $\mathcal{A}_1$, and eventually we will end up with a $\mathcal{A}_2'$ which makes no queries $\mathcal{O}_2$, in which case our discussion above shows untelegraphability security.

To this end, first, observe that the part of $\mathcal{A}_2$ until its first query to $\mathcal{O}_{\mathsf{OSS.GenQKey}}$ is executed can actually be considered as an adversary that does not have any queries to $\mathcal{O}_{\mathsf{OSS.GenQKey}}$ (after all, by definition it does not make any queries to this oracle before its *first* query to this oracle). Thus, we can actually invoke the incompressibility property for this part (call it $\mathcal{A}_2^{(1)}$). Due to the incompressibility property of OSS, we know that there is a small list $\mathfrak{L}$ such that when $\mathcal{A}_2'$ passes the verification of $\mathcal{O}_0$ and $\mathcal{O}_1$, it must be for some $vk \in \mathfrak{L}$. Thus, essentially, we can remove the access of $\mathcal{A}_2$ to the regions of the random oracle $H_0(vk||\cdot)$ and $H_1(vk||\cdot)$ for $vk \notin \mathfrak{L}$. Now, we know that during its query to $\mathcal{O}_2$, $\mathcal{A}_2$ will not have made any queries to $H_0(vk||\cdot), H_1(vk||\cdot)$ for $vk \notin \mathfrak{L}$. Then, we also observe that when $\mathcal{A}_2$ is querying $\mathcal{O}_2$, its valid queries cannot have $vk \in \mathfrak{L}$ since otherwise we actually show that we can extract a signature for both 0 and 1 with same vk from $\mathcal{A}_2$! Thus, we can invoke the incompressibility of random oracles H_0, H_1 outside the region $vk \notin \mathfrak{L}$ to say that valid queries of $\mathcal{A}_2$ to $\mathcal{O}_2$ must have z from a small set $\mathfrak{L}'$. Thus, before moving onto the next query, we can actually consider the outputs of $\mathcal{O}_2$ on these z values to be part of a new leakage from $\mathcal{A}_1'$ (who evaluates the oracle on this set and sends teh outputs), and thus remove the first query of $\mathcal{A}_2$ to $\mathcal{O}_2$ altogether and simulate it! One by one, we remove the queries of $\mathcal{A}_2$. Finally we show how to simulate the protocol between $\mathcal{A}_1$ and $\mathcal{A}_2$ using a single message and polynomially many queries by $\mathcal{A}_1^{sim}$ and no queries by $\mathcal{A}_2^{sim}$. The full proof is technical and deals with many other challenges (e.g., showing that we can extract a signature for 0 and 1 if valid query of $\mathcal{A}_2$ has $vk \in \mathfrak{L}$).

Another major point we swept under the rug in the discussion above is that, while we argued that we can remove oracle calls of $\mathcal{A}_2$ and replace them with simulations using classical messages from $\mathcal{A}_1'$, this actually requires knowing the small sets $\mathfrak{L}, \mathfrak{L}'$ on which $\mathcal{A}_2$ can actually *decrypt* the oracles. However, these existential sets (whose *existence* are guaranteed by the incompressibility properties of OSS and random oracles) are actually not available to use directly, so we cannot simply say that $\mathcal{A}_1'$ evaluates the oracles on these sets and sends the outputs to $\mathcal{A}_2'$. In general, these lists can depend on the full descriptions of the oracles and are not fully determined by the initial message L of $\mathcal{A}_1$, since $\mathcal{A}_1$ can use some outputs of the oracle as a one-time pad to encode other outputs,

[11] This is without loss of generality

and $\mathcal{A}_1$ and $\mathcal{A}_2$ can have a special encoding in their communication protocol. As another example, these adversaries can even use global properties of the oracles (such finding a preimage for a value using Yamakawa-Zhandry [YZ22]) in their encoding process. Thus, in general we will not have access to these sets, and giving these sets directly to the adversaries might break one-shot security (which we crucially rely on multiple points), or even break incompressibility property itself(!) since it is a global information about the oracles. To solve this issue, we show an estimation procedure where $\mathcal{A}_1$ estimates these sets by making polynomially many queries, and then uses the estimations, which we prove makes only a small difference. We refer the reader to Section 10 for details.

2.4 Interactive Security

In this work, we also initiate the study of *interactive* untelegraphability security for quantum fire. We first note that the previous definitions of untelegraphability security is very limited. In fact, while we do not show this explicitly, using our techniques it is easy to construct a scheme that satisfies (1-message) untelegraphability security, but its security breaks down fully when $\mathcal{A}_2$ first sends a message to $\mathcal{A}_1$ and $\mathcal{A}_1$ replies afterwards. Further, interactive adversaries better capture real-life leakage attacks: For example, when the flame states encode the secret key of a PKE scheme (see *key-fire* below), the adversary $\mathcal{A}_2$ will choose the leakage function depending on pk, which is not captured by previous definitions of untelegraphability.

Security Game We now give an overview of the interactive security game for mini-schemes, and the case of full schemes is defined similarly (see Definition 16 for the formal definition). The challenger samples a flame state as $sn^*, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QFireMini.Spark}(1^\lambda)$, and sends $sn^*, \mathsf{R}_{\mathsf{flame}}$ to $\mathcal{A}_1$. Then, $\mathcal{A}_1$ and $\mathcal{A}_2$ interact over a classical channel for any number of rounds. Finally, outputs a register R', which is tested for sn^* by running $\mathsf{QFireMini.Ver}(sn^*, \mathsf{R}')$.

Proving Security For interactive security, our construction stays the same as before, however, our proof is significantly different. Here we give a high-level overview, and but the full proof is highly delicate and technical - we refer the reader to Section 10 for details.

First, recall that our proof of untelegraphability security as discussed above crucially relied on running $\mathcal{A}_2$ twice on the message L from $\mathcal{A}_1$ to obtain two signatures, then we invoked incompressibility to obtain a contradiction to the one-shot security of OSS. However, once we move to the interactive setting, our argument completely breaks down: We can no longer run $\mathcal{A}_2$ twice since it has a (possibly unclonable) internal state, and incompressibility guarantees only apply to a non-interactive adversary $(\mathcal{A}_1, \mathcal{A}_2)$. To solve this, we show how to *clone* the internal state of the adversary $\mathcal{A}_2$ and also round-collapse the leakage protocol between $\mathcal{A}_1$ and $\mathcal{A}_2$ into a single message. To this end, we first start with our previous idea of making a fine-grained analysis of the queries of $\mathcal{A}_2$. We remove the queries of $\mathcal{A}_2$ to the oracles $\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2$ one by one and replace them with estimated ones using the incompressibility. Finally, after removing these calls, we go further and also show how to estimate and remove the calls of $\mathcal{A}_2$ to all the other oracles! Once we remove these calls, we are able to show that we decrease the number of rounds of interaction by making $\mathcal{A}_1$ simulate the first round, and then *telling* $\mathcal{A}_2$ the outcome of the first round, after which $\mathcal{A}_2$ locally prepares its internal state conditioned on that outcome/transcript so far! While normally it requires exponential many trials to correctly condition, since we have removed the queries of $\mathcal{A}_2$ (which are instead being simulated by simply reading from the database created by $\mathcal{A}_1$), we can actually afford to do this procedure. Finally once we collapse down to a single message from $\mathcal{A}_1$ to $\mathcal{A}_2$ with no actual oracle calls from $\mathcal{A}_2$, we can also clone the internal state of $\mathcal{A}_2$. Thus, we

are able to apply our previous argument based on one-shot security. The full proof deals with the challenges in making these arguments actually work, and is given in Section 10.

2.5 Quantum Key-Fire

In this work, we also introduce a new notion called *quantum key-fire*, where the flame states are *functional* and serve as a *key*. We give an overview, and the formal definitions can be found in Section 5.3. A quantum key-fire scheme consists of three algorithms: $\mathsf{QKeyFire.Protect}$, $\mathsf{QKeyFire.Eval}$, $\mathsf{QKeyFire.Clone}$. For a function f (modeled as a circuit), $\mathsf{Protect}(1^\lambda, f)$ outputs a flame state $|\psi_{pp,f}\rangle$ along with public parameters pp. $\mathsf{QKeyFire.Eval}(pp, |\psi_{pp,f}\rangle, x)$ evaluates the encoded functionality at x. Finally, $\mathsf{QKeyFire.Clone}(pp, |\psi_{pp,f}\rangle)$ clones the flame state. For correctness, we require that $\mathsf{Eval}(pp, |\psi_{pp,f}\rangle, x)$ outputs $f(x)$, and that $\mathsf{Clone}(pp, |\psi_{pp,f}\rangle)$ outputs $|\psi_{pp,f}\rangle \otimes |\psi_{pp,f}\rangle$. For security, we require that after an adversary obtains unbounded classical leakage on $|\psi_{pp,f}\rangle$ adaptively over any number of rounds, it still cannot correctly evaluate f at a challenge x^* (Definition 8).

Through known techniques ([CLLZ21, ALL+21, ÇG25]), it is easy to see that quantum key-fire for *signing* (Definition 18) implies quantum key-fire for any unlearnable functionality (in the classical oracle model). Thus, in the rest of the overview, we mainly consider quantum key-fire for signing. In the security game for this primitive (Definition 19), the adversary $\mathcal{A}$ receives LOCC leakage on the signing key state $\mathsf{R}_{\mathsf{flame}}$: For any number of rounds, it (adaptively) specifies a quantum circuit with classical output as a leakage function, which is then applied to $\mathsf{R}_{\mathsf{flame}}$ and the outcome is sent to $\mathcal{A}$. After the leakage, it is asked to sign a random message m^*. We require that any efficient adversary can win this game with negligible probability. Note that equivalently, this game can also be modeled as two adversaries $\mathcal{A}_1, \mathcal{A}_2$ interacting over a classical channel where $\mathcal{A}_1$ receives $\mathsf{R}_{\mathsf{flame}}$ and after the interaction, $\mathcal{A}_2$ receives the challenge m^* and produces a signature (see Definition 24 for this formalization).

Construction and Proving Security We briefly discuss our construction (we refer the reader to Section 9 for our full construction). The starting point of our construction is our key-fire construction, but with the addition of a new oracle that *verifies* the flame states and outputs shares of a random oracle evaluation $H_{\mathsf{sig}}(m)$ which serves a signature on m. In the proof, using the techniques discussed before, we show that we can round-collapse the leakage protocol down to a single message, and make the internal state of $\mathcal{A}_2$ clonable. At this point, we argue (thanks to incompressibility) that running two copies of $\mathcal{A}_2$ will query the evaluation oracle with the same OSS verification key, and we will be able to extract two OSS signatures to obtain a violation of the one-shot security. The full proof is highly delicate and technical - we refer the reader to Section 10.

2.6 Related Work

Nehoran and Zhandry ([NZ24]) did not formally define quantum fire, however, it introduced the complexity class $\mathsf{clonableQMA}$ and showed a quantum oracle separation between $\mathsf{clonableQMA}$ and QCMA. Note that $\mathsf{clonableQMA}$ sits between QMA and QCMA, thus a separation between $\mathsf{clonableQMA}$ and QCMA also leads to a separation between QMA and QCMA (note that such a separation was already known in the quantum oracle model). They also mentioned possible cryptographic applications, thus hinting at quantum fire. Indeed, their separation is also implicitly a quantum fire construction relative to the same quantum oracle, as observed by [BNZ25]. Unfortunately their techniques bake cloning into the oracle itself, and thus it uses an inherently inefficient quantum oracle and there is no hope even instantiating it even heuristically.

Bostanci, Nehoran, Zhandry ([BNZ25]) formalized the notion of quantum fire. Then, they gave a candidate construction based on group actions, and proved its cloning correctness. However, even in the classical oracle model, they only conjectured its security and could not give a proof of security or argue security. In fact, as discussed before, they even claimed that any provably secure quantum fire scheme in the classical oracle model would likely lead to a classical oracle separation between QMA and QCMA, which is one of the biggest questions in quantum cryptography that has been open since 2007. Indeed, since [NZ24] separates both clonableQMA and QCMA, and at the same time their separation is implicitly a quantum fire construction, it is an interesting question if there is any connection and if our techniques could be useful.

We also discuss a follow-up work, [HV25]. The initial version of our paper on arXiV contained a small gap in only in the last hybrid of our quantum key-fire proof (not quantum-fire proof), which was communicated to us by the authors of [HV25] and at the same time they put their paper online. Within 24 hours of receiving their communication, we fixed the small gap in our proof and let them know. Current version of our paper does not have this gap, however, we include a discussion in Section B for historical reference.

For leakage-resilience, [ÇGLZR24] introduced the notion of unbounded leakage-resilience and LOCC leakage-resilience for various cryptographic primitives such as public-key encryption and digital signatures. They also showed that the previous unclonable/copy-protected constructions of [CLLZ21, LLQZ22] also satisfy LOCC leakage-resilience. Therefore, they also asked the question of whether leakage-resilience security implies copy-protection security.

3 Preliminaries

In this section, we highlight some of our notation and conventions, and recall some basics about classical cryptography and quantum information.

3.1 Notation and Conventions

We refer the reader to [Gol01, Gol04] for a preliminary on cryptography and to [NC10] for a preliminary on quantum information and computation. We follow the notations and conventions commonly used in (quantum) cryptography, theoretical computer science and mathematics. In this section, we non-exhaustively highlight some of them. All of our conventions are implicitly followed unless otherwise specified, e.g. our cryptographic assumptions are always post-quantum even though most of the time we will not write this explicitly, and if there is a rare occasion where we assume classical security only, we will specify explicitly.

Functions For a function $f : \mathbb{N}^+ \to \mathbb{N}^+$, we will say that it is an efficiently computable function if it is polynomially bounded, that is, if there is some $n_0 \in \mathbb{N}^+$ and $c \in \mathbb{R}^+$ such that $f(n) \leq n^c$ for all $n > n_0$. Sometimes we will say polynomial to also mean polynomially bounded instead (e.g. when we say polynomial, it means $O(\mathsf{poly}(\lambda))$, which includes, say, $\sqrt{n}$) - the precise meaning will be clear from context. $f(\lambda)$ is said to be negligible if for any $c \in \mathbb{N}^+$, there is $n_0 \in \mathbb{N}^+$ such that $f(\lambda) \leq \frac{1}{n^c}$ for all $n \geq n_0$. $f(\lambda)$ is said to superlogarithmic if $f = \omega(\log(\lambda))$ or equivalently, if $2^{-f(\lambda)}$ is negligible.

When we describe a function, such as a polynomial, if no input is mentioned, it will implicitly be the security parameter λ.

Classical and Quantum Information and Oracles Variables written in lower-case, such as pk, sk, are classical objects. We write R to mean a quantum register, which keeps a quantum state that will evolve when the register is acted on, and it can be entangled with other registers.

We write $\mathsf{R} \leftarrow \rho$ to mean that the register R is initialized with a sample from the quantum distribution (i.e. mixed state) ρ. For simplicity, we will usually write non-normalized versions of the quantum states, however, they are always implicitly normalized.

We write $\mathcal{O}$ to denote a classical oracle, and write $\mathcal{A}^{\mathcal{O}}$ to denote an algorithm that has quantum query-access to $\mathcal{O}$. Similarly, for unitary oracles we write U and $\mathcal{A}^U$. All adversaries always have quantum access to all of the oracles.

Distributions When S is a set, we write $x \leftarrow S$ to mean that x is sampled uniformly at random from S. When $\mathcal{D}$ is a distribution, we write $x \leftarrow \mathcal{D}$ to mean that x is sampled from $\mathcal{D}$. Finally, we write $x \leftarrow \mathcal{B}(\mathsf{R})$ or $x \leftarrow \mathcal{B}(a)$ to mean that x is a sample as output by the (quantum or classical randomized) algorithm $\mathcal{B}$ run on the quantum input register R or classical input a. We use similar notation for quantum registers, e.g., $\mathsf{R} \leftarrow \mathcal{B}(a)$.

Adversaries and Constructions/Algorithms We write *QPT* to mean *quantum polynomial time* and *PPT* to mean *probabilistic (classical) polynomial time*. We say query-bounded to mean that the algorithm (e.g. construction, adversary) makes polynomially many queries to the oracle, but it can possibly take unbounded time outside the queries. All of these are polynomial in the security parameter λ. Note that for QPT and PPT, this also means that the inputs to these algorithms are also of polynomial size in λ.

We say *efficient* to mean QPT or PPT (will be clear from context) in the plain model, or query-bounded in the oracle model. For constructions, efficient means QPT or PPT, even in the oracle model (though the oracles themselves might not be efficiently sampleable or implementable)

In the plain model, adversaries will be implicitly QPT. In the oracle security model, adversaries will be implicitly *query-bounded*. An admissible adversary means an adversary that conforms to the (implicit) requirements of the security game (e.g. polynomial time or query bounded, has the correct interaction model and has correct input-output format and so on). All of our cryptographical assumptions are post-quantum, e.g., *one-way functions* means *post-quantum secure one-way functions*. Adversaries are stateful uniform QPT.

When we say that a primitive is subexponentially secure, we mean that any polynomial time adversary has subexponentially small advantage. Sometimes we require security against subexponential time adversaries, in which case we will specify this explicitly.

All algorithms of a cryptographic scheme are uniform (stateless) PPT or QPT (will be clear from context).

Security Games A security game is a binary random variable which is the outcome of an experiment between a *challenger* and an adversary. For a security game, we say that the adversary has won if the experiment output is 1, and otherwise we say that the adversary has lost. When we say that an adversary has negligible advantage, we mean that the probability of the adversary winning (or distinguishing, depending on context) is upper bounded by a negligible function of λ.

Domains We write $\mathcal{M}$ to denote the message space for various cryptographic primitives, and it will be equal to $\{0,1\}^{p(\lambda)}$ for some polynomial $p(\cdot)$ (will be clear from context).

3.2 Pseudorandom Functions

In this section, we recall pseudorandom functions (PRF) and puncturable PRFs.

Definition 2 (Pseudorandom Functions)**.** *A pseudorandom function (PRF) scheme* PRF *is a family of functions* $\{F : \{0,1\}^{c(\lambda)} \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}\}_{\lambda\in\mathbb{N}^+}$ *along with the following efficient algorithms.*

- $\mathsf{PRF.Gen}(1^\lambda)$: *Takes in the unary representation of the security parameter and outputs a key* K *in* $\{0,1\}^{c(\lambda)}$*.*
- $\mathsf{PRF.Eval}(K,x)$: *Takes in a key* $K \in \{0,1\}^{c(\lambda)}$ *and an input* $x \in \{0,1\}^{m(\lambda)}$*, outputs an evaluation of the PRF in* $\{0,1\}^{n(\lambda)}$*.*

We require the following.

Correctness.

$$\Pr\left[\forall x\ \mathsf{PRF.Eval}(K,x) = F(K,x) :\ \ K \leftarrow \mathsf{PRF.Setup}(1^\lambda)\ \right] = 1.$$

Security *We require that for any QPT adversary* $\mathcal{A}$*,*

$$\left|\Pr_{H\leftarrow \mathrm{H}}[\mathcal{A}^H(1^\lambda)=1] - \Pr_{K\leftarrow\mathsf{PRF.Gen}(1^\lambda)}[\mathcal{A}^{F(K,\cdot)}(1^\lambda)=1]\right| \le \mathsf{negl}(\lambda).$$

where H *is the uniform distribution over the functions* $\{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}$*.*

We will usually write $F_K(x)$ or $F(K,x)$ in algorithms and this will implicitly mean $\mathsf{PRF.Eval}(K,x)$.

Definition 3 (Puncturable Pseudorandom Functions [SW14])**.** *A puncturable pseudorandom function (PRF) scheme* PRF *is a pseudorandom function scheme along with the following additional efficient algorithm.*

- $\mathsf{PRF.Puncture}(K,S)$: *Takes as input a key* $K \in \{0,1\}^{c(\lambda)}$ *and a predicate circuit* $C : \{0,1\}^{m(\lambda)} \to \{0,1\}$*, outputs a punctured key.*

We require the following.

Punctured Correctness. *For all efficient distributions* $\mathcal{D}(1^\lambda)$ *over predicate circuits, we require*

$$\Pr\left[\forall x \in \{0,1\}^{m(\lambda)} C(x) = 1 \implies \mathsf{PRF.Eval}(K_C, x) = F(K,x) : \begin{array}{c} C \leftarrow \mathcal{D}(1^\lambda) \\ K \leftarrow \mathsf{PRF.Setup}(1^\lambda) \\ K_C \leftarrow \mathsf{PRF.Puncture}(K, C) \end{array}\right] = 1.$$

Puncturing Security *We require that any QPT adversary* $\mathcal{A}$ *wins the following game with probability at most* $1/2 + \mathsf{negl}(\lambda)$*.*

1. *The adversary* $\mathcal{A}$ *outputs a predicate circuit* C*.*
2. *The challenger samples* $K \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ *and* $K_C \leftarrow \mathsf{PRF.Puncture}(K, C)$*, and submits* K_C *to the adversary* $\mathcal{A}$*.*
3. *The adversary* $\mathcal{A}$ *outputs an input* x *such that* $C(x) = 0$*.*
4. *The challenger samples* $b \leftarrow \{0,1\}$*. If* $b = 0$*, the challenger submits* $F(K,x)$ *to the adversary. Otherwise, it submits* y *to the adversary where* $y \leftarrow \{0,1\}^{n(\lambda)}$*.*
5. *The adversary outputs a guess* b' *and we say that the adversary has won if* $b' = b$*.*

It is easy to see that any puncturable PRF scheme that satisfies puncturing security also satisfies usual PRF security.

Theorem 5 ([SW14, Zha12])**.** *Let* $n(\cdot), m(\cdot)$ *be polynomially bounded.*

- *If (post-quantum) one-way functions exist, then there exists a (post-quantum) puncturable PRF with input space* $\{0,1\}^{m(\lambda)}$ *and output space* $\{0,1\}^{n(\lambda)}$*.*
- *If subexponentially-secure (post-quantum) one-way functions exist, then for any* $c > 0$*, there exists a (post-quantum)* $2^{-\lambda^c}$*-secure*[12] *puncturable PRF with input space* $\{0,1\}^{m(\lambda)}$ *and output space* $\{0,1\}^{n(\lambda)}$*.*

[12]While the original results are for negligible security against polynomial time adversaries, it is easy to see that they carry over to subexponential security. Further, by scaling the security parameter by a polynomial and simple input/output conversions, subexponentially secure (for any exponent c') one-way functions is sufficient to construct for any c a puncturable PRF that is $2^{-\lambda^c}$-secure.

3.3 Digital Signature Schemes

In this section we recall the definition of signatures schemes.

Definition 4. *A digital signature scheme with message space $\mathcal{M}$ consists of the following efficient algorithms that satisfy the correctness and security guarantees below.*

- $\mathsf{Setup}(1^\lambda)$: *Takes the unary representation of the security parameter and outputs a signing key sk and a verification key vk.*
- $\mathsf{Sign}(sk, m)$: *Takes a signing key sk and a message $m \in \mathcal{M}$, returns a signature for m.*
- $\mathsf{Ver}(vk, m, sig)$: *Takes the public verification key vk, a message $m \in \mathcal{M}$ and an alleged signature sig for m, outputs 1 if s is a valid signature for m.*

Correctness *We require the following for all messages $m \in \mathcal{M}$.*

$$\Pr\left[\mathsf{Ver}(vk, m, s) = 1 : \begin{array}{c} sk, vk \leftarrow \mathsf{Setup}(1^\lambda) \\ s \leftarrow \mathsf{Sign}(sk, m) \end{array}\right] = 1.$$

Adaptive existential-unforgeability security under chosen message attack (EUF-CMA) *We require that any QPT adversary $\mathcal{A}$ wins the following game with negligible probability.*

1. *The challenger samples the keys $sk, vk \leftarrow \mathsf{Setup}(1)$.*
2. *The adversary $\mathcal{A}$ receives vk, and interacts with the signing oracle by sending classical messages $m \in \mathcal{M}$ and receiving the corresponding signatures that is computed by the challenger as $sig \leftarrow \mathsf{Sign}(sk, m)$.*
3. *The adversary $\mathcal{A}$ outputs a message $m^* \in \mathcal{M}$ that it has not queried the oracle with and a forged signature sig^*.*
4. *The challenger outputs 1 if and only if $\mathsf{Ver}(vk, m^*, sig^*) = 1$.*

If the adversary $\mathcal{A}$ is required to output the message m^* before receiving vk, we call it selective EUF-CMA *security.*

3.4 Indistinguishability Obfuscation

In this section, we recall indistinguishability obfuscation (iO).

Definition 5 (Indistinguishability Obfuscation)**.** *An indistinguishability obfuscation (iO) scheme for a class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ is an efficient algorithm $i\mathcal{O}$ that satisfies the following.*

Correctness. *For all $\lambda \in \mathbb{N}^+, C \in \mathcal{C}_\lambda$ and all inputs x to C,*

$$\Pr\left[\tilde{C}(x) = C(x) : \tilde{C} \leftarrow i\mathcal{O}(1^\lambda, C)\right] = 1.$$

Security. *Let $\mathcal{B}$ be any QPT algorithm that outputs two circuits $C_0, C_1 \in \mathcal{C}$ of the same size, along with quantum auxiliary information $\mathsf{R}_{\mathsf{aux}}$, such that $\Pr\left[\forall x\ C_0(x) = C_1(x) : (C_0, C_1, \mathsf{R}_{\mathsf{aux}}) \leftarrow \mathcal{B}(1^\lambda)\right] \geq 1 - \mathsf{negl}(\lambda)$. Then, for any QPT adversary $\mathcal{A}$,*

$$\left| \Pr\left[\mathcal{A}(i\mathcal{O}(1^\lambda, C_0), \mathsf{R}_{\mathsf{aux}}) = 1 : (C_0, C_1, \mathsf{R}_{\mathsf{aux}}) \leftarrow \mathcal{B}(1^\lambda)\right] - \Pr\left[\mathcal{A}(i\mathcal{O}(1^\lambda, C_1), \mathsf{R}_{\mathsf{aux}}) = 1 : (C_0, C_1, \mathsf{R}_{\mathsf{aux}}) \leftarrow \mathcal{B}(1^\lambda)\right] \right| \leq \mathsf{negl}(\lambda).$$

3.5 Quantum Information

In this section, we recall some quantum information results.

We first recall the gentle measurement lemma, which says that if some measurement has a high probability of giving a particular outcome on some state, then the measurement can be implemented in a way so that the state will not be disturbed too much when that outcome occurs.

Lemma 1 (Gentle Measurement Lemma [Win99])**.** *Let ρ be a mixed state acting on $\mathbb{C}^d$. Let U be a unitary and $(\Pi_0, \Pi_1 = I - \Pi_0)$ be projectors all acting on $\mathbb{C}^d \otimes \mathbb{C}^{d'}$. We interpret (U, Π_0, Π_1) as a measurement performed by appending an ancillary system of dimension d' in the state $|0\rangle\langle 0|$, applying U and then performing the projective measurement Π_0, Π_1 on the larger system. Assuming that the outcome corresponding to Π_0 has probability $1 - \epsilon$, we have*

$$\left|\rho - \rho'\right| \leq \sqrt{\epsilon}$$

where ρ' is the state after performing the measurement, undoing the unitary U and tracing out the ancillary system.

We recall the *hybrid lemma* for reprogramming quantumly-accessible classical oracles.

Theorem 6 (Hybrid Lemma [BBBV97])**.** *Let $\mathcal{A}$ be a quantum algorithm making queries to an oracle $\mathcal{O}$. Let $|\psi_t\rangle = \sum_{w,x,y,t} \alpha_{w,x,y,t} |w, x, y\rangle$ denote the joint state of the working register, the query input register and the query output register of the algorithm right before execution of the t-th query. For a subset S of the domain of $\mathcal{O}$, let $q_S(|\psi_t\rangle) = \sum_{x \in S} |\alpha_{w,x,y,t}|^2$ and $q_S = \sum_t q_s(|\psi_t\rangle)$, and call q_S the query weight of S. Let $\mathcal{O}'$ be another oracle whose output differs from $\mathcal{O}$ only on points $x \in S$. Then, if $\mathcal{A}$ makes T queries to the oracle $\mathcal{O}$ and S is a subset such that $q_S \leq \epsilon^2/T$, we have $||\psi\rangle\langle\psi| - |\psi'\rangle\langle\psi'||_1 \leq 2\epsilon$ where $|\psi\rangle, |\psi'\rangle$ denote the final state of the algorithm $\mathcal{A}$ when given access to the oracles $\mathcal{O}, \mathcal{O}'$ respectively. In the case where $\mathcal{O}, \mathcal{O}', S$ are distributional (along with some auxiliary information register $\mathsf{R}_{\mathsf{aux}}$) and possibly correlated, we have $(|\psi\rangle\langle\psi|, \mathcal{O}, \mathcal{O}', S, \mathsf{R}_{\mathsf{aux}}) \approx_\epsilon (|\psi'\rangle\langle\psi'|, \mathcal{O}, \mathcal{O}', S, \mathsf{R}_{\mathsf{aux}})$ or equivalently $(|\psi\rangle\langle\psi|, \mathcal{O}, \mathcal{O}', S, \mathsf{R}_{\mathsf{aux}}) \approx_{\sqrt{q_S \cdot T}} (|\psi'\rangle\langle\psi'|, \mathcal{O}, \mathcal{O}', S, \mathsf{R}_{\mathsf{aux}})$.*

We recall the following lemma about compressed oracle simulation technique for random oracles. We will use this technique in Section 7, thus, we refer the reader to [Zha19] for details of this technique. As a reminder of the notation, we have $\mathsf{CStO} = \mathsf{StdDecomp} \circ \mathsf{CStO}' \circ \mathsf{StdDecomp} \circ \mathsf{Increase}$.

Theorem 7 (Compressed Oracle Lemma [Zha19])**.** *Consider a quantum algorithm $\mathcal{A}$ making queries to a random oracle $H : \{0,1\}^m \to \{0,1\}^n$ and outputting a tuple $(x_1, \ldots, x_k, y_1, \ldots, y_k, z)$. Let R be a collection of such tuples. Suppose with probability p, $\mathcal{A}$ outputs a tuple such that (1) the tuple is in R and (2) $H(x_i) = y_i$ for all i. Now consider running the* compressed oracle simulation *of $\mathcal{A}$, suppose the database D is measured after $\mathcal{A}$ produces its output. Let p' be the probability that (1) the tuple is in R, and (2) $D(x_i) = y_i$ for all i (and in particular $D(x_i) \neq \perp$). Then $\sqrt{p} \leq \sqrt{p'} + \sqrt{\frac{k}{2^n}}$.*

We recall the small-range distributions lemma.

Theorem 8 (Small Range Distributions [Zha12])**.** *There is a universal constant $C_{\mathsf{SRD}} > 1$ such that, for any sets $\mathcal{X}$ and $\mathcal{Y}$, distribution $\mathcal{D}$ on $\mathcal{Y}$, any integer ℓ, and any quantum algorithm $\mathcal{A}$ that makes k queries, we have*

$$\left| \Pr_{\mathcal{O} \leftarrow \mathfrak{O}_1} [\mathcal{A}^{\mathcal{O}}() = 1] - \Pr_{\mathcal{O} \leftarrow \mathfrak{O}_2} [\mathcal{A}^{\mathcal{O}}() = 1] \right| \leq C_{\mathsf{SRD}} \frac{k^3}{\ell}$$

where the distributions $\mathfrak{O}_1, \mathfrak{O}_2$ are defined as follows.

- *For $\mathfrak{O}_1$, independently sample $y_x \leftarrow \mathcal{D}$ for each $x \in \mathcal{X}$ and set $\mathcal{O}(x) = y_x$.*
- *For $\mathfrak{O}_2$, sample a random function $P : \mathcal{X} \to [\ell]$, independently sample $y_x \leftarrow \mathcal{D}$ for each $x \in [\ell]$ and set $\mathcal{O}(x) = y_{P(x)}$.*

Part I
Definitions

4 Quantum Protection Definitions

In this section, we recall the definitions of various *quantum protection* ([ÇG24, ÇG25]) security notions, which is the question of encoding a key/functionality into a quantum state to obtain a classically-impossible security guarantee (such as copy-protection or unbounded leakage-resilience).

4.1 Quantum Protection

We recall the notions of classically unlearnable functionalities and quantum protection schemes.

Definition 6 (Classically Unlearnable Functionalities [ÇG24, ÇG25])**.** *A functionality $\mathcal{F}$ is a tuple $(F,$ $\mathsf{SampFunc}$, $\mathsf{SampChal}$, $\mathsf{Pred})$ whose elements are defined as follows.*

- *F: A sequence consisting of sets (for each $\lambda \in \mathbb{N}^+$) of classical circuits of polynomial size.*
- $\mathsf{SampFunc}(1^\lambda)$: *An interactive efficient (quantum) algorithm that outputs a value $f \in F$ at the end of the interaction.*
- $\mathsf{SampChal}(f)$*: An efficient (quantum) algorithm that takes as input a value $f \in F$ and outputs a challenge input x and an answer key ak.*
- $\mathsf{Pred}(f, ak, x, y)$*: An efficient (quantum) algorithm that takes as input a value $f \in F$, an answer key ak, an input x and a value y, and outputs 1 or 0, denoting if y is a valid answer for input x with respect to f and ak or not.*

Consider the following between a challenger and an adversary $\mathcal{A}$.

$\underline{\mathsf{LEARNING-GAME}_{\mathcal{A}}(1^\lambda)}$

1. *The challenger and the adversary interact where the challenger simply simulates* $\mathsf{SampFunc}$*. Let f^* denote the output of* $\mathsf{SampFunc}$ *at the end of the interaction.*
2. *For any polynomial number of rounds: The adversary $\mathcal{A}$ submits the classical description of a oracular quantum circuit C with classical output, the challenger executes C with oracle access to f^* and submits the outcome to the adversary.*
3. *The challenger samples a challenge input $x^*, ak^* \leftarrow \mathsf{SampChal}(f^*)$ and submits x^* to the adversary $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs y^*.*
5. *The challenger outputs 1 if and only if $\mathsf{Pred}(f^*, ak^*, x^*, y^*)$ outputs 1.*

A functionality $\mathcal{F}$ is said to satisfy p_{triv}-classical-unlearnability *if for any QPT adversary $\mathcal{A}$,*

$$\Pr\left[\mathsf{LEARNING-GAME}_{\mathcal{A}}(1^\lambda) = 1\right] \leq p_{triv}(\lambda) + \mathsf{negl}(\lambda).$$

Note that for any non-trivial game (i.e. p_{triv} is not close to 1), the size of the challenge x^* must be superlogarithmic.

Definition 7 (Quantum Protection Scheme [ÇG24, ÇG25])**.** *Let* $\mathcal{F} = (F, \mathsf{SampFunc}, \mathsf{SampChal}, \mathsf{Pred})$ *be a functionality. A quantum protection scheme* QuantumProtection *for* $\mathcal{F}$ *consists of the following efficient algorithms.*

- $\mathsf{QuantumProtection.Protect}(1^\lambda, f)$*: On input a circuit* $f \in F$*, outputs a quantum register* $\mathsf{R}_{\mathsf{key}}$*.*
- $\mathsf{QuantumProtection.Eval}(\mathsf{R}_{\mathsf{key}}, x)$*: On input a key register and a circuit input* x*, outputs an evaluation.*

We require evaluation correctness: For each $f \in F$ *and input* x *to* f*,*

$$\Pr\left[y = f(x) : \begin{array}{c} \mathsf{R}_{\mathsf{key}} \leftarrow \mathsf{QuantumProtection.Protect}(1^\lambda, f) \\ y \leftarrow \mathsf{QuantumProtection.Eval}(\mathsf{R}_{\mathsf{key}}, x) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

4.2 LOCC Leakage-Resilience

We recall LOCC leakage-resilience security for general functionalities.

Definition 8 (LOCC Leakage-Resilience Security [ÇGLZR24, ÇG24, ÇG25])**.** *Let* $\mathcal{F} = (F,$ SampFunc, SampChal, Pred$)$ *be a functionality that is* p_{triv}*-classical-unlearnable and let* QuantumProtection *be a quantum protection scheme for* $\mathcal{F}$*. Consider the following game between the challenger and an adversary* $\mathcal{A}$*.*

$\underline{\mathsf{LEAK} - \mathsf{GAME}_{\mathcal{A}}(1^\lambda)}$

1. *The challenger and the adversary* $\mathcal{A}$ *interact where the challenger simply simulates* SampFunc*. Let* f^* *denote the output of* SampFunc *at the end of the interaction.*
2. *The challenger samples* $\mathsf{R}_0 \leftarrow \mathsf{QuantumProtection.Protect}(1^\lambda, f^*)$*.*
3. ***Leakage Phase:*** *Then, for any number*[13] *of rounds, the following is executed (starting with* $i = 1$*).*
    1. $\mathcal{A}$ *specifies a quantum leakage circuit* E_i *that takes as input a quantum register and outputs a classical string and the updated quantum register.*
    2. *The challenger executes* $L_i, \mathsf{R}_i \leftarrow E_i(\mathsf{R}_{i-1})$ *and submits the classical string* L_i *to* $\mathcal{A}$*. Then* i *is incremented.*
4. *The challenger samples a challenge input* $x^* \leftarrow \mathsf{SampChal}(f^*)$ *and submits* x^* *to the adversary* $\mathcal{A}$*.*
5. $\mathcal{A}$ *outputs* y^**.*
6. *The challenger outputs* 1 *if and only if* $\mathsf{Pred}(f^*, ak^*, x^*, y^*)$ *outputs* 1*.*

A quantum protection scheme QuantumProtection *is said to satisfy* LOCC leakage-resilience *for the functionality* $\mathcal{F}$ *if for any QPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\mathsf{LEAK} - \mathsf{GAME}_{\mathcal{A}}(1^\lambda) = 1\right] \leq p_{triv}(\lambda) + \mathsf{negl}(\lambda).$$

[13] Note that when $\mathcal{A}$ is QPT, implicitly this will be any *polynomial* number of rounds.

Note that previous works also define unbounded (i.e. 1-round) leakage-resilience. In this work, since we achieve the strongest notion, LOCC leakage-resilience, we will not define unbounded leakage-resilience.

In Section A, for ease of exposition, we also explicitly write out the special case of this definition for digital signatures and public-key encryption explicitly, since these are the two simplest and most natural use cases.

4.3 One-Shot Signatures

We recall one-shot signatures.

Definition 9 (One-Shot Signatures [AGKZ20])**.** *A one-shot signature scheme* OSS *consists of the following efficient algorithms.*

- $\mathsf{OSS.Setup}(1^\lambda)$*: Takes in a security parameter, outputs a common reference string* crs*.*
- $\mathsf{OSS.GenQKey}(crs)$*: Takes in the CRS value* crs*, outputs a verification key* vk *and a quantum signing key* $\mathsf{R}_{\mathsf{key}}$*.*
- $\mathsf{OSS.Sign}(crs, vk, \mathsf{R}_{\mathsf{key}}, m)$*: Takes in the CRS value,a verification key, a signing key* $\mathsf{R}_{\mathsf{key}}$*, and a message* m*; outputs a classical signature* sig*.*
- $\mathsf{OSS.Ver}(crs, vk, m, sig)$*: Takes in the CRS value* crs*, a verification key* vk*, a message* m *and a signature* sig*,; outputs* 0 *or* 1*.*

Correctness: *For any message* $m \in \mathcal{M}$*,*

$$\Pr\left[out_{\mathsf{Ver}} = 1 : \begin{array}{c} crs \leftarrow \mathsf{OSS.Setup}(1^\lambda) \\ vk, \mathsf{R}_{\mathsf{key}} \leftarrow \mathsf{OSS.GenQKey}(crs) \\ sig \leftarrow \mathsf{OSS.Sign}(crs, vk, \mathsf{R}_{\mathsf{key}}, m) \\ out_{\mathsf{Ver}} \leftarrow \mathsf{OSS.Ver}(crs, vk, m, sig) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

One-Shot Security: *For any QPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{c} m_1 \neq m_2 \\ \wedge \\ out_{1,\mathsf{Ver}} = 1 \\ \wedge \\ out_{2,\mathsf{Ver}} = 1 \end{array} : \begin{array}{c} crs \leftarrow \mathsf{OSS.Setup}(1^\lambda) \\ vk, sig_1, m_1, sig_2, m_2 \leftarrow \mathcal{A}(crs) \\ out_{1,\mathsf{Ver}} \leftarrow \mathsf{OSS.Ver}(crs, vk, m_1, sig_1) \\ out_{2,\mathsf{Ver}} \leftarrow \mathsf{OSS.Ver}(crs, vk, m_2, sig_2) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

In Section 6, we introduce a new security notion called incompressibility for one-shot signature in the classical oracle model. Thus, we also define OSS in the oracle model below[14].

Definition 10 (One-Shot Signatures in the Classical Oracle Model)**.** *A one-shot signature scheme* OSS *in the classical oracle model consists of the following efficient algorithms.*

- $\mathsf{OSS.GenOracles}(1^\lambda)$*: Takes in a security parameter, outputs a classical oracle* $\mathcal{O}$*.*

[14]This is slightly different from the standard oracle model where there is a single global oracle (rather than an oracle being output for each instance of the scheme) that all the algorithms and adversaries query. We use this model for simplicity, and they are equivalent to each other.

- $\mathsf{OSS.GenQKey}^{(\cdot)}(1^\lambda)$*: An oracle algorithm that takes in a security parameter, outputs a verification key* vk *and a quantum signing key* $\mathsf{R_{key}}$*.*
- $\mathsf{OSS.Sign}^{(\cdot)}(vk, \mathsf{R_{key}}, m)$*: An oracle algorithm that takes in a verification key, a signing key* $\mathsf{R_{key}}$*, and a message* m*; outputs a classical signature* sig*.*
- $\mathsf{OSS.Ver}^{(\cdot)}(vk, m, sig)$*: An oracle algorithm that takes in a verification key* vk*, a message* m *and a signature* sig*,; outputs* 0 *or* 1*.*

Correctness: *For any message* $m \in \mathcal{M}$*,*

$$\Pr\left[out_{\mathsf{Ver}} = 1 : \begin{array}{c} \mathcal{O} \leftarrow \mathsf{OSS.GenOracles}(1^\lambda) \\ vk, \mathsf{R_{key}} \leftarrow \mathsf{OSS}^{\mathcal{O}}.\mathsf{GenQKey}(1^\lambda) \\ sig \leftarrow \mathsf{OSS.Sign}^{\mathcal{O}}(vk, \mathsf{R_{key}}, m) \\ out_{\mathsf{Ver}} \leftarrow \mathsf{OSS.Ver}^{\mathcal{O}}(vk, m, sig) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

One-Shot Security: *For any query-bounded adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{c} m_1 \neq m_2 \\ \wedge \\ out_{1,\mathsf{Ver}} = 1 \\ \wedge \\ out_{2,\mathsf{Ver}} = 1 \end{array} : \begin{array}{c} \mathcal{O} \leftarrow \mathsf{OSS.GenOracles}(1^\lambda) \\ vk, sig_1, m_1, sig_2, m_2 \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda) \\ out_{1,\mathsf{Ver}} \leftarrow \mathsf{OSS}^{\mathcal{O}}.\mathsf{Ver}(vk, m_1, sig_1) \\ out_{2,\mathsf{Ver}} \leftarrow \mathsf{OSS}^{\mathcal{O}}.\mathsf{Ver}(vk, m_2, sig_2) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

We define strong one-shot security similarly, with the difference that it is only required to have $sig_1 \neq sig_2$ *(and not necessarily* $m_1 \neq m_2$*).*

5 Definitional Work on Quantum Fire

In this section, we recall some existing definitions related to quantum fire, and also introduce definitions for new primitives and security notions.

5.1 Quantum Fire

We first recall quantum fire ([BNZ25]). Throughout the paper, we will only consider public-key quantum fire even when we ignore writing out *public-key* explicitly.

Definition 11 (Quantum Fire)**.** *A (public-key) quantum fire scheme* QFire *consists of the following efficient algorithms.*

- $\mathsf{QFire.Setup}(1^\lambda)$*: Takes in a security parameter, outputs a public key* pk *and a secret key* sk*.*
- $\mathsf{QFire.Spark}(sk)$*: Takes in the secret key, outputs a serial number* sn*, and a* flame state *in the register* $\mathsf{R}_{\mathsf{flame}}$*.*
- $\mathsf{QFire.Clone}(pk, sn, \mathsf{R}_{\mathsf{flame}})$*: Takes in the public key, a serial number, a flame state register* $\mathsf{R}_{\mathsf{flame}}$*, outputs a new flame register* $\mathsf{R}_{\mathsf{clone}}$ *(along with the input register* $\mathsf{R}_{\mathsf{flame}}$*).*
- $\mathsf{QFire.Ver}(pk, sn, \mathsf{R})$*: Takes in the public key, a serial number, and an alleged flame state, outputs* 0 *or* 1*.*

We require the following correctness guarantees.

Verification Correctness:

$$\Pr\left[out_{\mathsf{Ver}} = 1 : \begin{array}{c} pk, sk \leftarrow \mathsf{QFire.Setup}(1^\lambda) \\ sn, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QFire.Spark}(sk) \\ out_{\mathsf{Ver}} \leftarrow \mathsf{QFire.Ver}(pk, sn, \mathsf{R}_{\mathsf{flame}}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Cloning Correctness: *For any polynomial* $p(\cdot)$*,*

$$\Pr\left[\forall i \in [p(\lambda)]\ out_{i,\mathsf{Ver}} = 1 : \begin{array}{c} pk, sk \leftarrow \mathsf{QFire.Setup}(1^\lambda) \\ sn, \mathsf{R}_1 \leftarrow \mathsf{QFire.Spark}(sk) \\ \textit{For } i = 2, \ldots, p(\lambda)\ \mathsf{R}_i \leftarrow \mathsf{QFire.Clone}(pk, sn, \mathsf{R}_{i-1}) \\ \textit{For } i = 1, \ldots, p(\lambda)\ out_{i,\mathsf{Ver}} \leftarrow \mathsf{QFire.Ver}(pk, sn, \mathsf{R}_i) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

We introduce a stronger correctness condition called *strong cloning correctness.*

Definition 12 (Strong Cloning Correctness)**.** *A quantum fire scheme* QFire *is said to satisfy* strong cloning correctness *if the following are satisfied: There exists a collection of pure states* $|\psi_{sk,sn}\rangle$ *such that with probability* 1 *the output of* $\mathsf{QFire.Spark}(sk)$ *is* $sn, |\psi_{sk,sn}\rangle$ *for some* sn*. Let* $M_{sk,sn}$ *denote the binary outcome projective measurement onto* $|\psi_{sk,sn}\rangle$*.*

$$\Pr\left[b_1 = b_2 = 1 : \begin{array}{c} pk, sk \leftarrow \mathsf{QFire.Setup}(1^\lambda) \\ sn, \mathsf{R}_{flame} \leftarrow \mathsf{QFire.Spark}(sk) \\ \mathsf{R}_{clone} \leftarrow \mathsf{QFire.Clone}(pk, sn, \mathsf{R}_{flame}) \\ b_1 \leftarrow M_{sk,sn}(\mathsf{R}_{flame}) \\ b_2 \leftarrow M_{sk,sn}(\mathsf{R}_{clone}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Note that in oracle-based constructions, the flame states also depend on the oracle itself[15] *in addition to* sk, sn.

We now recall untelegraphability security for quantum fire. We slightly extend the definition of [BNZ25] to consider collusion-resistant quantum fire, where the adversary gets multiple independent flame states.

Definition 13 (Untelegraphability Security)**.** *Consider the following game between an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *and a challenger.*

$\underline{\mathsf{TELEGRAPH} - \mathsf{GAME}_{\mathcal{A}}(1^\lambda)}$

1. *The challenger samples* $pk, sk \leftarrow \mathsf{QFire.Setup}(1^\lambda)$ *and submits* pk *to* $\mathcal{A}_1$ *and* $\mathcal{A}_2$.
2. *For polynomially many number of rounds: The adversary sends an empty query, and the challenger samples* $sn, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QFire.Spark}(sk)$ *and submits* $sn, \mathsf{R}_{\mathsf{flame}}$ *to the adversary* $\mathcal{A}_1$.
3. *The adversary* $\mathcal{A}_2$ *outputs a message* T.
4. *The adversary* $\mathcal{A}_1$ *receives* T *and outputs a classical string* L.
5. *The adversary* $\mathcal{A}_2$ *receives* L*, outputs* sn^*, R'.
6. *The challenger outputs* 1 *if* $b = 1$ *where* $b \leftarrow \mathsf{QFire.Ver}(pk, sn^*, \mathsf{R}') = 1$*, and otherwise it outputs* 0.

We say that the quantum fire scheme QFire *satisfies* untelegraphability security *if for any for any polynomial-time (or query-bounded, for unconditional oracle-based schemes) adversary* $\mathcal{A}$*, we have*

$$\Pr\left[\mathsf{TELEGRAPH} - \mathsf{GAME}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

Note that the weaker notion of *non-adaptive untelegraphability security* can also be defined, where $\mathcal{A}_1$ produces its output without receiving a message from $\mathcal{A}_2$.

Finally, we introduce the notion of mini-scheme for quantum fire, analogous to the notion of mini-scheme for quantum money. This definition makes proofs easier, and it is easy to see that any mini-scheme for quantum fire can generically be upgraded to a full scheme using digital signatures (Definition 4), simply by signing the serial number (signature is made part of the new serial number) and verifying the signature on the serial number as part of verification - just like in the case of upgrading quantum money mini-schemes to full schemes ([AC12]).

Definition 14 (Quantum Fire Mini-Scheme)**.** *A public-key quantum fire mini-scheme* QFireMini *consists of the following efficient algorithms.*

- $\mathsf{QFireMini.Spark}(1^\lambda)$*: Takes in the security parameter, outputs a serial number* sn*, and a* flame state $\mathsf{R}_{\mathsf{flame}}$.
- $\mathsf{QFireMini.Clone}(sn, \mathsf{R}_{\mathsf{flame}})$*: Takes in a serial number, a flame register, outputs a new flame register* $\mathsf{R}_{\mathsf{clone}}$ *(along with the input register* $\mathsf{R}_{\mathsf{flame}}$*).*
- $\mathsf{QFireMini.Ver}(sn, \mathsf{R})$*: Takes in a serial number, and an alleged flame state, outputs* 0 *or* 1.

[15] Equivalently, we can consider the whole oracle as part of the secret key for purposes of this definition.

Verification Correctness:

$$\Pr\left[out_{\mathsf{Ver}} = 1 : \begin{array}{c} sn, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QFireMini.Spark}(1^\lambda) \\ out_{\mathsf{Ver}} \leftarrow \mathsf{QFireMini.Ver}(sn, \mathsf{R}_{\mathsf{flame}}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Strong Cloning Correctness: *We require that there exists a collection of pure states $|\psi_k\rangle$ such that with probability 1 the output of* $\mathsf{QFireMini.Spark}(1^\lambda)$ *is $sn, |\psi_k\rangle$ for some k, sn and*

$$\Pr\left[b_1 = b_2 = 1 : \begin{array}{c} sn, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QFireMini.Spark}(1^\lambda) \\ \mathsf{R}_{clone} \leftarrow \mathsf{QFireMini.Clone}(sn, \mathsf{R}_{flame}) \\ b_1 \leftarrow M_k(\mathsf{R}_{flame}) \\ b_2 \leftarrow M_k(\mathsf{R}_{clone}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

where M_k denotes the binary outcome projective measurement onto $|\psi_k\rangle$.

Note that in oracle-based constructions, the flame states also depend on the oracle itself[16] *in addition to k.*

Definition 15 (Untelegraphability Security for Quantum Fire Mini-Scheme)**.** *Consider the following game between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger.*

$\underline{\mathsf{MINI} - \mathsf{TELEGRAPH} - \mathsf{GAME}_{\mathcal{A}}(1^\lambda)}$

1. *The challenger samples $sn, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QFireMini.Spark}(1^\lambda)$ and submits $sn, \mathsf{R}_{\mathsf{flame}}$ to the adversary $\mathcal{A}_1$.*
2. *The adversary $\mathcal{A}_2$ receives sn and outputs a message T.*
3. *The adversary $\mathcal{A}_1$ receives T and outputs a classical string L.*
4. *The adversary $\mathcal{A}_2$ receives L, outputs R'.*
5. *The challenger executes $\mathsf{QFireMini.Ver}(sn, \mathsf{R}')$ outputs if it outputs 1, otherwise it outputs 0.*

We say that the quantum fire scheme QFireMini *satisfies* untelegraphability security *if for any QPT (or query bounded, in case of unconditional oracle-based schemes) adversary $\mathcal{A}$, we have*

$$\Pr\left[\mathsf{MINI} - \mathsf{TELEGRAPH} - \mathsf{GAME}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

5.2 Interactive Untelegraphability Security

We introduce the notion of interactive telegraphing security, inspired by the notion of *LOCC leakage-resilience* (Definition 8) defined by [ÇGLZR24]. For simplicity, we will explicitly define it only for mini-schemes, but the collusion-resistant case is defined similarly to Definition 13 (where $\mathcal{A}_1$ receives multiple independent flame states).

Definition 16 (Interactive Untelegraphability Security)**.** *Consider the following game between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger.*

[16] Equivalently, we can consider the whole oracle as part of the key k for purposes of this definition.

$\mathsf{MINI-INTERACTIVE-TELEGRAPH-GAME}_{\mathcal{A}}(1^\lambda)$

1. *The challenger samples* $sn, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QFireMini.Spark}(1^\lambda)$ *and submits* $sn, \mathsf{R}_{\mathsf{flame}}$ *to the adversary* $\mathcal{A}_1$.
2. *The adversaries* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *interact classically by exchanging classical messages for any number of rounds*[17].
3. *The adversary* $\mathcal{A}_2$ *receives* L*, outputs* R'.
4. *The challenger executes* $\mathsf{QFireMini.Ver}(sn, \mathsf{R}')$ *outputs if it outputs* 1*, otherwise it outputs* 0.

We say that the quantum fire scheme QFireMini *satisfies* interactive untelegraphability security *if for any QPT (or query bounded, in case of unconditional oracle-based schemes) adversary* $\mathcal{A}$*, we have*

$$\Pr\left[\mathsf{MINI-INTERACTIVE-TELEGRAPH-GAME}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

5.3 Quantum Key-Fire

We now introduce a new notion called *quantum key-fire*. Quantum key-fire is the functional version of quantum fire, where the flame state can be used to evaluate a functionality, and we require *unbounded/LOCC leakage-resilience* ([ÇGLZR24]) as our security notion, which is a generalization of the notion of untelegraphability. Quantum key-fire is to quantum fire what copy-protection ([ALL+21, CLLZ21]) is to public-key quantum money. Just like the case of copy-protection and quantum money, it is easy to see that quantum key-fire implies quantum fire.

A quantum key-fire scheme is an *quantum protection scheme* (Definition 7) with unbounded/LOCC leakage-resilience, with the addition of a cloning algorithm. We write the full definition for reference.

Definition 17 (Quantum Key-Fire)**.** *A public-key quantum key-fire scheme for a functionality* $\mathcal{F}$ *(Definition 6) consists of the following efficient algorithms.*

- $\mathsf{QKeyFire.Protect}(1^\lambda, f)$*: Takes in a function* $f \in F$ *and outputs a* flame state $\mathsf{R}_{\mathsf{key}}$.
- $\mathsf{QKeyFire.Clone}(\mathsf{R}_{\mathsf{key}})$*: Takes in a flame register* $\mathsf{R}_{\mathsf{key}}$*, outputs a new register* $\mathsf{R}_{\mathsf{clone}}$ *(along with the input register* $\mathsf{R}_{\mathsf{key}}$*).*
- $\mathsf{QKeyFire.Eval}(\mathsf{R}_{\mathsf{key}}, x)$ *: Takes in a flame register* $\mathsf{R}_{\mathsf{key}}$*, and an input* x*, outputs a value.*

Evaluation Correctness: *For all* $f \in F$*, for all* x*,*

$$\Pr\left[y = f(x) : \begin{array}{c} \mathsf{R}_{\mathsf{key}} \leftarrow \mathsf{QKeyFire.Protect}(1^\lambda, f) \\ y \leftarrow \mathsf{QKeyFire.Eval}(\mathsf{R}_{\mathsf{key}}, x) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Strong Cloning Correctness: *We require that there exists a collection of pure states* $|\psi_{k,f}\rangle$ *such that for all* $f \in F$*, with probability* 1 *the output of* $\mathsf{QKeyFire.Protect}(1^\lambda, f)$ *is* $|\psi_{k,f}\rangle$ *for some* k *and*

$$\Pr\left[b_1 = b_2 = 1 : \begin{array}{c} k, sn, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QKeyFire.Protect}(1^\lambda, f) \\ \mathsf{R}_{clone} \leftarrow \mathsf{QKeyFire.Clone}(\mathsf{R}_{flame}) \\ b_1 \leftarrow M_{k,f}(\mathsf{R}_{flame}) \\ b_2 \leftarrow M_{k,f}(\mathsf{R}_{clone}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

[17] Note that when $\mathcal{A}_1, \mathcal{A}_2$ are QPT, implicitly this will be any *polynomial* number of rounds.

where $M_{k,f}$ denotes the binary outcome projective measurement onto $|\psi_{k,f}\rangle$.

Note that in oracle-based constructions, the flame states also depend on the oracle itself[18] *in addition to f, k.*

Security: *When* QKeyFire *is a key-fire scheme for some (classically-unlearnable) functionality $\mathcal{F}$ (Definition 6), we will require that* QKeyFire *satisfies LOCC leakage-resilience (Definition 8) for $\mathcal{F}$.*

5.3.1 Quantum Key-Fire for Signing

In this section, we provide the explicit definition of the special case of quantum-key fire for signing, since this the main functionality we consider in this work.

Definition 18 (Quantum Key-Fire for Signing)**.** *A public-key quantum key-fire scheme for signing* QKeyFireSign *consists of the following efficient algorithms.*

- $\mathsf{QKeyFireSign.Setup}(1^\lambda)$*: Takes in a security parameter, outputs a verification key (also called serial number) vk, and a* key state $\mathsf{R_{flame}}$.
- $\mathsf{QKeyFireSign.Clone}(vk, \mathsf{R_{key}})$*: Takes in a key register $\mathsf{R_{key}}$, outputs a new register $\mathsf{R_{clone}}$ along with the input register $\mathsf{R_{key}}$.*
- $\mathsf{QKeyFireSign.Sign}(vk, \mathsf{R_{key}}, m)$ *: Takes in a key register and a message m, outputs a signature.*
- $\mathsf{QKeyFireSign.Ver}(vk, m, sig)$*: Takes in a verification key (also called serial number), a message and an alleged signature, outputs* 1 *or* 0*.*

Evaluation Correctness: *For all $m \in \mathcal{M}$,*

$$\Pr\left[\mathsf{QKeyFireSign.Ver}(vk, m, sig) = 1 : \begin{array}{c} vk, \mathsf{R_{key}} \leftarrow \mathsf{QKeyFireSign.Setup}(1^\lambda) \\ sig \leftarrow \mathsf{QKeyFireSign.Sign}(vk, \mathsf{R_{key}}, m) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Strong cloning correctness: *This is defined the same as before: We require that output of $\mathsf{QKeyFireSign.Setup}(1^\lambda)$ is a pure state $|\psi_k\rangle$ and that clone on input $|\psi_k\rangle$ outputs two copies, $|\psi_k\rangle \otimes |\psi_k\rangle$.*

Definition 19 (LOCC Leakage-Resilience Security for Quantum Key-Fire for Signing)**.** *Consider the following game between an adversary $\mathcal{A}$ and a challenger.*

$\underline{\mathsf{LR-SIGN-GAME}_{\mathcal{A}}(1^\lambda)}$

1. *The challenger samples $vk, \mathsf{R_{key}} \leftarrow \mathsf{QKeyFireSign.Setup}(1^\lambda)$ and submits vk to $\mathcal{A}$.*
2. *Set $\mathsf{R}_0 = \mathsf{R_{key}}$.*
3. ***Leakage Phase:** For any number*[19] *of rounds, the following is executed (starting with $i = 1$).*
    1. *$\mathcal{A}$ specifies a quantum leakage circuit E_i that takes as input a quantum register and outputs a classical string and the updated quantum register.*

[18] Equivalently, we can consider the whole oracle as part of the key k for purposes of this definition.

[19] Note that when $\mathcal{A}$ is QPT, implicitly this will be any *polynomial* number of rounds.

2. *The challenger executes* $L_i, \mathsf{R}_i \leftarrow E_i(\mathsf{R}_{i-1})$ *and submits the classical string* L_i *to* $\mathcal{A}$*. Then* i *is incremented.*

4. *The challenger samples a random message* $m^* \leftarrow \mathcal{M}$*.*

5. *The adversary* $\mathcal{A}$ *receives* m^**, outputs* sig^**.*

6. *The challenger checks if* $\mathsf{QKeyFireSign.Ver}(sn, m^*, sig^*) = 1$*. If so, it outputs* 1*, otherwise it outputs* 0*.*

We say that the quantum key-fire scheme QKeyFireSign *satisfies* LOCC leakage-resilience security *if for any QPT (or query bounded, in case of unconditional oracle-based schemes) adversary* $\mathcal{A}$*, we have*

$$\Pr\Big[\mathsf{LR-SIGN-GAME}_{\mathcal{A}}(1^\lambda) = 1\Big] \leq \mathsf{negl}(\lambda).$$

Part II
Technical Tools

6 One-Shot Signatures with Incompressibility Security

In this section, we introduce a new notion called *incompressibility* security for one-shot signatures in the classical oracle model, and then construct a scheme that satisfies it. The section is organized as follows. First we introduce our definition of incompressibility and give our theorem statement about existence of an incompressible OSS scheme. Then in Section 6.1, we introduce a technical tool called *operational incompressibility* that simplifies the proof of our incompressibility notion. Then we give our construction in Section 6.2 and prove its incompressibility property in Section 6.4. In Section 6.3, we show some technical results related to subspace oracles that will be used in our proof in Section 6.4 and might be of independent interest.

For our incompressibility notion, we consider a one-shot signature scheme OSS with *split oracles.*

Definition 20 (One-Shot Signature Scheme with Split Oracles)**.** *Let* OSS *be a one-shot signature scheme with classical oracles (Definition 10). Write* $\mathcal{O}_{\mathsf{OSS}}$ *to denote its oracle.* OSS *is said to have* split oracles *if* $\mathcal{O}_{\mathsf{OSS}}$ *is a tuple of (correlated) oracles* $(\mathcal{O}_{\mathsf{OSS.GenQKey}}, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}) = \mathcal{O}_{\mathsf{OSS}}$*, where each algorithm of* OSS *uses only the corresponding oracle (e.g.* $\mathsf{OSS.GenQKey}$ *only uses* $\mathcal{O}_{\mathsf{OSS.GenQKey}}$*). We also assume that the inputs to* $\mathcal{O}_{\mathsf{OSS.Sign}}$ *and* $\mathcal{O}_{\mathsf{OSS.Ver}}$ *are of the form* vk, x *where* vk *denotes a verification key.*

Note that any one-shot signature scheme can trivially be converted into a scheme with split oracles by setting all three oracles to be the full oracle, but this will not satisfy incompressibility.

Now we define incompressibility.

Definition 21 (Incompressibility)**.** OSS *is said to satisfy* incompressibility security *for* $\mathcal{O} \in \{\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}\}$ *if the following is satisfied. There exists some* $c > 0$ *such that for any query-bounded adversary* $\mathcal{A}_1$ *with unbounded length classical output, there exists a mapping* $\mathcal{M}$ *(which outputs a list*[20] $\mathfrak{L}$ *with at most* $s(\lambda)$ *elements where* $s(\lambda) = (q(\lambda))^c$ *and* $q(\lambda)$ *is the number of* $\mathcal{O}_{\mathsf{OSS.GenQKey}}$ *queries made by* $\mathcal{A}_1$*) such that for all query-bounded* $\mathcal{A}_2$*, we have*

$$\mathbb{E}\left[\sum_{\substack{vk,x:\\ \mathcal{O}(vk,x)\neq 0\\ vk\notin\mathcal{M}(\mathcal{O}_{\mathsf{OSS}},L,\mathcal{A}_2)}} \mathsf{QW}^{(vk,x)}_{\mathcal{A}_2,L,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_{\mathsf{OSS.Ver}}} : \begin{array}{c}\mathcal{O}_{\mathsf{OSS}}\leftarrow\mathsf{OSS.GenOracles}(1^\lambda)\\ L\leftarrow\mathcal{A}_1^{\mathcal{O}_{\mathsf{OSS}}}(1^\lambda)\end{array}\right]\leq \mathsf{negl}(\lambda).$$

where $\mathsf{QW}^{(vk,x)}_{\mathcal{A}_2,L,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_{\mathsf{OSS.Ver}}}$ *denotes the total query weight of* $\mathcal{A}_2^{\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_{\mathsf{OSS.Ver}}}(L)$ *to the oracle* $\mathcal{O}$ *on oracle input* vk, x*, and* $\mathcal{A}_1, \mathcal{A}_2$ *are allowed to share an unbounded length classical randomness in advance (i.e. before oracles are sampled).*

If incompressibility is satisfied for both $\mathcal{O} \in \{\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}\}$*, then we will simply say that OSS satisfies incompressibility.*

We define *strong incompressibility* similarly, where we require the list $\mathcal{M}$ to output pairs (vk, z) and the summation above is taken over $(vk, z) \notin \mathcal{M}(\mathcal{O}_{\mathsf{OSS}}, L, \mathcal{A}_2)$. It is easy to see that strong

[20] We will refer to this list as the incompressibility list.

incompressibility for $\mathcal{O}_{\mathsf{OSS.Ver}}$ follows from incompressibility for $\mathcal{O}_{\mathsf{OSS.Ver}}$ and strong one-shot security.

We claim that a one-shot signature scheme with incompressibility security exists.

Theorem 9. *Relative to a classical oracle, there exists a one-shot signature scheme for 1-bit messages that satisfies one-shot security and incompressibility security.*

We give our construction in Section 6.2 and prove its incompressibility in Section 6.4.

6.1 Operational Incompressibility

As a technical tool for proving incompressibility, we define the notion of operational incompressibility.

Definition 22 (Operational Incompressibility)**.** OSS *is said to satisfy* operational incompressibility security *for* $\mathcal{O} \in \{\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}\}$ *if the following is satisfied. There exists some* $c > 0$ *such that for any query-bounded adversary* $\mathcal{A}_1$ *with unbounded-length classical output, for all query-bounded* $\mathcal{A}_2$*, we have*

$$\Pr\left[\begin{array}{c} \forall i \in [s(\lambda)+1]\ \mathcal{O}(vk_i, x_i) \neq 0 \\ \wedge \\ (vk_i)_{i\in[s(\lambda)+1]}\ \textit{distinct} \end{array} : \begin{array}{c} \mathcal{O}_{\mathsf{OSS}} \leftarrow \mathsf{OSS.GenOracles}(1^\lambda) \\ L \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{OSS}}}(1^\lambda) \\ (vk_i, x_i)_{i\in[s(\lambda)+1]} \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}}(L) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

where $s(\lambda) = (q(\lambda))^c$ *and* $q(\lambda)$ *is the number of* $\mathcal{O}_{\mathsf{OSS.GenQKey}}$ *queries made by the adversary* $\mathcal{A}_1$ *and* $\mathcal{A}_1, \mathcal{A}_2$ *are allowed to share an unbounded length classical randomness in advance (i.e. before oracles are sampled).*

Theorem 10. *If an OSS scheme satisfies operational incompressibility, then it also satisfies incompressibility.*

Now we prove this theorem. Let OSS be a scheme that satisfies operational incompressibility. Fix $\mathcal{O} \in \{\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}\}$. For any vk, define

$$q_{vk} = \sum_{x:\mathcal{O}(vk,x)\neq 0} \mathsf{QW}^{vk,x}_{\mathcal{A}_2, L, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}}$$

The dependence on $\mathcal{A}_2, L, \mathcal{O}_{\mathsf{OSS}}$ are implicit.

Let $vk_1, \ldots, vk_s, vk_{s+1}$ be the inputs such that

- $q_{vk_i} \geq q_{vk_{i+1}}$ for $i \in [s]$
- $q_{vk_{s+1}} \geq q_{vk'}$ for any $vk' \notin \{vk_i\}_{i\in[s+1]}$

That is, vk_i are the inputs with the highest *valid* query weights. Note that these all implicitly depend on $\mathcal{O}_{\mathsf{OSS}}, L, \mathcal{A}_2$.

We set $\mathcal{M}(\mathcal{O}_{\mathsf{OSS}}, L, \mathcal{A}_2)$ to be $\{vk_1, \ldots, vk_s\}$. Now suppose for a contradiction that incompressibility property is not satisfied, that is, there exists some $\mathcal{A}_1, \mathcal{A}_2$ and polynomial $t(\cdot)$ such that for infinitely many $\lambda > 0$,

$$p_{out} = \mathbb{E}\left[\sum_{\substack{vk: \\ vk\notin\mathcal{M}(\mathcal{O}_{\mathsf{OSS}}, L, \mathcal{A}_2)}} q_{vk} : \begin{array}{c} \mathcal{O}_{\mathsf{OSS}} \leftarrow \mathsf{OSS.GenOracles}(1^\lambda) \\ L \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{OSS}}}(1^\lambda) \end{array}\right] \geq \frac{1}{t(\lambda)}.$$

Let $u(\lambda)$ denote the number of queries to $\mathcal{O}$ by $\mathcal{A}_2$. Consider the following adversary $\mathcal{A}_2'$ for the operational incompressibility game.

$\underline{\mathcal{A}'_2}$

1. Set $D = \emptyset$.
2. For $n(\lambda) = u(\lambda) \cdot t(\lambda) \cdot s(\lambda) \cdot \lambda^2 \cdot (s(\lambda) + 1)$ times (in $s + 1$ *groups*):
   1. Simulate $\mathcal{A}_2(L)$, but measure a random query to $\mathcal{O}$ to obtain vk, x.
   2. If $\mathcal{O}(vk, x) \neq 0$ and $vk \notin D$[21], add (vk, x) to D and skip to the next *group*.
3. Output D.

We will show that we can violate operational incompressibility using $\mathcal{A}'_2$. We continue our proof by considering two cases/events. Define E_1 to be the event that $q_{vk_{s+1}} \geq \frac{1}{t(\lambda)\cdot s(\lambda)\cdot\lambda}$, and E_2 to be its complement. Thus we have either $\Pr[E_1] \geq \frac{1}{4\cdot t\cdot u-2}$ or $\Pr[E_2] \geq 1 - \frac{1}{4\cdot t\cdot u-2}$.

Now we consider the case $\Pr[E_1] \geq \frac{1}{4\cdot t\cdot u-2}$. We claim that $\mathcal{A}_1, \mathcal{A}'_2$ wins the operational incompressibility game with probability at least $\frac{1}{8\cdot t\cdot u-4}$. We condition on the event E_1, and we will show that the adversaries will win with probability $\geq 1 - \mathsf{negl}(\lambda)$. Observe that for each vk_i with $i \in [s+1]$, we have that the probability of $vk_i \in D$ is $1 - \exp\{-\frac{n}{u\cdot t\cdot s\cdot\lambda}\} \geq 1 - \exp\{-\lambda\}$ since the weight of each vk_i is at least $\frac{1}{t\cdot s\cdot\lambda}$. Thus, by the union bound, we get that $vk_1, \ldots, vk_{s+1} \in D$ with overwhelming probability, which indeed violates incompressibility security.

Now we consider the case $\Pr[E_2] \geq 1 - \frac{1}{4\cdot t\cdot u-2}$. Let GOOD be the event that

$$\sum_{vk:vk\notin\mathcal{M}(\mathcal{O}_{\mathsf{OSS}},L,\mathcal{A}_2)} q_{vk} \geq \frac{1}{2\cdot t}$$

Since expectation of this value (p_{out}) is greater than $\frac{1}{t(\lambda)}$ and since $\Pr[E_2] \geq 1 - \frac{1}{4\cdot t\cdot u-2}$, we get $\Pr[E_2 \wedge \mathsf{GOOD}] \geq \frac{1}{4\cdot t\cdot u-2}$ by an application of Markov's inequality and union bound.

We claim that $\mathcal{A}_1, \mathcal{A}'_2$ wins the operational incompressibility game with overwhelming probability conditioned on $E_2 \wedge \mathsf{GOOD}$, which happens with non-negligible probability. Let G denote the complement set of $\mathcal{M}(\mathcal{A}_2, \mathcal{O}_{\mathsf{OSS}}, L)$. Let $vk^*_1, \ldots, vk^*_{s+1}$ be the values added to the database D during each group during execution of $\mathcal{A}'_2$, in the order of addition. In particular, if no item was added during a group, we set $vk^*_i = \perp$. We now claim that with overwhelming probability, $vk^*_1, \ldots, vk^*_{s+1} \neq \perp$. Note that once we establish this, we will have shown that we can violate operational incompressibility with non-negligible probability. Now to prove our claim, first observe that $vk^*_1 \in G$ with overwhelming probability since $q_G := \sum_{vk\in G} q_{vk} \geq \frac{1}{2t}$ (since we conditioned on GOOD). Then, we claim that $vk^*_2 \neq \perp$ with overwhelming probability. To see this, observe that $q_G \geq \frac{1}{2t}$ and for any possible value of $vk^*_1 \in G$, we have $q_{vk^*_1} \leq \frac{1}{t\cdot s\cdot\lambda}$ (since we conditioned on E_2) and thus $q_G - q_{vk^*_1} \geq \frac{1}{2t}$. Continuing like this, we can show that each vk^*_i is $\in G$ with overwhelming probability, no matter what the previous values $vk^*_1, \ldots, vk^*_{i-1}$ are. This completes case 2.

Observe that we showed violations of the operational incompressibility game in both cases. This completes the proof.

6.2 Construction and Proofs of Security

In this section, we prove Theorem 9: We show that there exists a one-shot signature scheme that satisfies incompressibility (Definition 21) relative to a classical oracle. Our construction will be a delicately modified version of the construction of [SZ25].

[21] Here we mean that there is no x such that $(vk, x) \in D$

Now we move onto our construction.

Define the following parameters: Let $q(\lambda) = 16\lambda, r(\lambda) = q(\lambda)\cdot(\lambda-1), n(\lambda) = r(\lambda)+\frac{3}{2}q(\lambda), k(\lambda) = n(\lambda)$.

$\underline{\mathsf{OSS.GenOracles}(1^\lambda)}$

1. Sample a random permutation π on $\{0,1\}^n$. Let $H(x), J(x)$ denote the first r and last $n-r$ bits of $\pi(x)$.
2. For each $y \in \{0,1\}^r$, sample a random full column-rank matrix $A_y \in \mathbb{F}_2^{k\times(n-r)}$ and a random vector b_y in $\mathbb{F}_2^k$.
3. Set $\mathcal{P}(x) = (y, A_y \cdot J(x) + b_y)$ where $y = H(x)$.
4. Set $\mathcal{P}^{-1}(y, v) = \pi^{-1}(y||z)$ if $v = A_y \cdot z + b_y$ for some $z \in \mathbb{F}_2^{n-r}$, otherwise $\perp$.
5. Set $\mathcal{D}(y, v) = 1$ if $v^T A_y = 0$ and $v \neq 0$, otherwise 0.
6. Set $\mathcal{D}_0(y, m, v) = 1$ if $v = A_y \cdot z + b_y$ for some $z \in \mathbb{F}_2^{n-r}$ and the first bit of v is m; otherwise 0.
7. **The oracles:** Set $\mathcal{O}_{\mathsf{Ver}} = \mathcal{D}_0$, $\mathcal{O}_{\mathsf{Sign}} = \mathcal{D}$, $\mathcal{O}_{\mathsf{GenQKey}} = (\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}, \mathcal{D}_0)$. Set the oracle $\mathcal{O}_{\mathsf{OSS}}$ to be $(\mathcal{O}_{\mathsf{Ver}}, \mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{GenQKey}})$.

$\underline{\mathsf{OSS.GenQKey}^{\mathcal{O}_{\mathsf{GenQKey}}}(1^\lambda)}$

1. Parse $\mathcal{O}_{\mathsf{GenQKey}} = (\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}, \mathcal{D}_0)$.
2. Prepare the states $|0^n\rangle, |0^r\rangle, |0^k\rangle$ over the registers $\mathsf{R}_1, \mathsf{R}_2, \mathsf{R}_3$.
3. Apply $H^{\otimes n}$ to R_1.
4. Execute the oracle $\mathcal{P}$ with input register R_1 and output register $(\mathsf{R}_2, \mathsf{R}_3)$.
5. Execute the oracle $\mathcal{P}^{-1}$ with input register $(\mathsf{R}_2, \mathsf{R}_3)$ and output register R_1.
6. Measure the register R_2, let y be the measurement outcome.
7. Output y, R_3.

$\underline{\mathsf{OSS.Ver}^{\mathcal{O}_{\mathsf{Ver}}}(vk, sig, m)}$

1. Parse $\mathcal{O}_{\mathsf{Ver}} = \mathcal{D}_0$.
2. Output $\mathcal{D}_0(vk, m, sig) = 1$.

Our signing algorithm is the same as the signing algorithm of [SZ25] (which in turn is the same as the signing mechanism of [AGKZ20, Shm22]). For completeness, we recall it here.

$\underline{\mathsf{OSS.Sign}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{R}_{\mathsf{key}}, m)}$

1. Parse $\mathcal{O}_{\mathsf{Sign}} = \mathcal{D}$.
2. Parse $(vk, \mathsf{R}'_{\mathsf{key}}) = \mathsf{R}_{\mathsf{key}}$
3. For $j = 1, \ldots, \lambda$:
    1. Measure the first qubit of $\mathsf{R}'_{\mathsf{key}}$. In case the outcome is m, measure the rest of $\mathsf{R}'_{\mathsf{key}}$ to obtain sig', and output $m||sig'$ and terminate.
    2. Apply $H^{\otimes k}$ to $\mathsf{R}'_{\mathsf{key}}$, then apply the following mapping coherently (writing the outcome to a new output register) to $\mathsf{R}'_{\mathsf{key}}$ and measure the output bit: $(vk, z) \to 1$ if $\mathcal{D}(vk, z) = 1$ or if $z = 0^k$, otherwise $(vk, z) \to 0$. Then, apply $H^{\otimes k}$ again to $\mathsf{R}'_{\mathsf{key}}$. Proceed to the next iteration $j \leftarrow j + 1$.
4. If all λ iterations failed, output $\perp$.

Theorem 11. OSS *satisfies correctness.*

Proof. While the correctness of the scheme follows from previous work ([SZ25], which in turn uses the correctness proof of [AGKZ20, Shm22]), for ease of reference, we recall it below. Fix any $m \in \{0, 1\}$ for the rest of the proof.

First, we assume that for all but (at most) $2 \cdot 2^{2r(\lambda)-n(\lambda)}$ many values of y, we have that A_y has at least a single column whose first entry is 1. Since the probability of this happening for any individual y value is $1 - (1/2)^{n-r}$, by a simple Chernoff bound it is easy to see that our assumption holds with probability at least $1 - \exp\{-\lambda\}$.

It is easy to see that $\mathsf{GenQKey}$ outputs $y, \sum_{u \in \mathsf{ColSpan}(A_y)} |v + b_y\rangle$ for a uniformly random y, thus we start with this state in the register $\mathsf{R}_{\mathsf{key}}$. We assume that A_y has at least a single column whose first entry is 1. Note that we had already assumed that this is already true for all but (at most) $2 \cdot 2^{2r(\lambda)-n(\lambda)}$ many verification keys, and since y is chosen uniformly at random from 2^r verification keys, our assumption holds with probability at least $1 - \exp\{-\lambda\}$.

Note that by definition signatures for m correspond to vectors in $\mathsf{ColSpan}(A_y) + b_y$ that start with the bit m. Since there is at least one column whose first entry is 1, it is easy to see that there is an equal number of vectors that start with either bit in $\mathsf{Colspan}(A_y) + b_y$. Thus, the probability of signing in the first iteration of the algorithm is accordingly $1/2$. The algorithm makes λ tries and we would like to argue it succeeds with probability $1 - 2^{\lambda}$. We will prove that each try indeed succeeds with at least probability $1/2$ (so far it is clear only for the first iteration), which will complete the proof.

Now let $S := \mathsf{ColSpan}(A_y)$ and let S_0 the set of vectors in S that start with 0, which is a subspace of S with one less dimension. Observe that after the failed first try, the state we have in $\mathsf{R}'_{\mathsf{key}}$ is the state $\sum_{u \in S_0} |b_y + x + u\rangle$, for some value $x \in S$ which will not matter for our proof (for the interested reader, x will be the zero vector if b_y starts with the bit $\overline{m}$, and otherwise it will be a vector that *completes* S_0 to S). Now we claim that that if we have a state of the form $\sum_{u \in S_0} (-1)^{\langle z,u \rangle} |b_y + x + u\rangle$, for any $z \in S_0^{\perp}$, after performing the correction part of the signing algorithm, we obtain a state of the form $\sum_{u \in S} (-1)^{\langle z',u \rangle} |b_y + x + u\rangle$ for some $z' \in S_0^{\perp}$. To see this, note that after executing $H^{\otimes k}$ the state we get is $\sum_{v \in S_0^{\perp}} (-1)^{\langle b_y + x, v \rangle} |z + v\rangle$, and since S_0 is a subspace of S with one less dimension, the subspace $S^{\perp} := \mathsf{ColSpan}(A_y)^{\perp}$ is a subspace of $S_0^{\perp}$ with one less dimension. Thus, when we measure the output qubit of $\mathcal{D}$ (note that we are not using the output of $\mathcal{D}$ and effectively discarding its answer) the state collapses to a state of the form

$\sum_{v\in S^\perp}(-1)^{\langle b_y+x,v\rangle}|z'+v\rangle$ for some value $z'\in S_0^\perp$. When we execute the second Hadamard, we get a state of the form $\sum_{u\in S}(-1)^{\langle z',u\rangle}|b_y+x+u\rangle$, which is equal to $\sum_{u\in S}(-1)^{\langle z',u\rangle}|b_y+u\rangle$ since $x\in S$. Thus, in summary, as long as we start an iteration with a state of this form, we end the iteration with a state of this form (even the signing attempt fails). With this, it is easy to see that each iteration succeeds with probability 1/2 independently. This finishes the proof. □

Theorem 12. OSS *satisfies strong one-shot security.*

Proof. Observe that the oracle $\mathcal{D}_0$ in our construction can be implemented using $\mathcal{P}^{-1}$. Since strong one-shot security relative to $\mathcal{P},\mathcal{P}^{-1},\mathcal{D}$ has already been proven by [SZ25], the security of our scheme follows. □

Theorem 13. OSS *satisfies incompressibility.*

We prove this theorem in Section 6.4.

6.3 Properties of Subspace Oracles

In this section, we recall some results about subspace oracles and also prove some new technical results. Throughout the section, we will write $\mathcal{O}_{S'}$ to denote the oracle that checks membership in S' (outputs 1 if and only if the input is inside S') where S' is a subspace.

First we recall the subspace-hiding oracles.

Lemma 2 (Subspace-Hiding Oracles [SZ25])**.** *Let $k,r,s\in\mathbb{N}$ be such that $r+s\le k$ and let S be a subspace of $\mathbb{F}_2^k$ of dimension r. Let $\mathcal{U}$ be the uniform distribution over superspaces T of S of dimension $r+s$.*

Then, for any adversary $\mathcal{A}$ making at most q quantum queries, we have the following.

$$\{\mathcal{A}^{\mathcal{O}_S}()\}\approx_{O\left(\frac{q\cdot s}{\sqrt{2^{k-r-s}}}\right)}\{\mathcal{A}^{\mathcal{O}_T}()\ :\ T\leftarrow\mathcal{U}\}\ .$$

Now we prove the following theorem, which significantly improves upon a similar result of [SZ25].

Theorem 14. *Let $d_2\le d_1\le n$. Let S be a subspace of dimension d_1 of $\mathbb{F}_2^n$, and let $\mathcal{U}$ denote the uniform distribution of subspaces of S of dimension d_2. Then, for any adversary $\mathcal{A}$ making at most q quantum queries, we have*

$$\Pr\left[v\in T\setminus\{0\}:\ \begin{matrix}T\leftarrow\mathcal{U}\\ v\leftarrow\mathcal{A}^{\mathcal{O}_{T^\perp}}\end{matrix}\right]\le\frac{2^{d_1-d_2}-1}{2^{d_1}-1}+q\cdot\sqrt{\frac{2^{d_1-d_2}-1}{2^{d_1}-1}}$$

Proof. Let $p_{\mathcal{A}}=\Pr\left[v\in T\setminus\{0\}:\ \begin{matrix}T\leftarrow\mathcal{U}\\ v\leftarrow\mathcal{A}^{\mathcal{O}_{T^\perp}}\end{matrix}\right]$. We observe that for any adversary $\mathcal{B}$ that is not given query access to $\mathcal{O}_{T^\perp}$ (call such an adversary an *oblivious adversary*), the probability of outputting a vector $v\in T^\perp\setminus S^\perp$ is $p=\frac{2^{d_1-d_2}-1}{2^{d_1}-1}$. Thus, by Theorem 6, if we modify the first oracle call of $\mathcal{A}$ to instead use $\mathcal{O}_{S^\perp}$, the probability of winning decreases by at most $\sqrt{p}$, since the part of the adversary before its first call to $\mathcal{O}_{T^\perp}$ can be considered an oblivious adversary. Further, once we modify the first oracle call of the adversary to $\mathcal{O}_{S^\perp}$, now the part until the second oracle call is also an oblivious adversary. Continuing this argument for all q oracle calls, we can conclude that when $\mathcal{A}$ is run with the oracle $\mathcal{O}_{S^\perp}$, its probability of winning is at least $p_{\mathcal{A}}-q\cdot\sqrt{p}$. However, by the same oblivious algorithm argument as above, we have $p_{\mathcal{A}}-q\cdot\sqrt{p}\le p$. Thus, $p_{\mathcal{A}}\le q\cdot\sqrt{p}+p$. □

6.4 Proof of Incompressibility

In this section, we prove Theorem 13. In the proof, instead of setting $s(\lambda) = (q(\lambda))^c$ for some $c > 0$, we actually set $s(\lambda) = 2q(\lambda)$, which is even stronger.

Now, we move onto our proof. We will explicitly prove only the case of $\mathcal{O} = \mathcal{O}_{\mathsf{OSS.Ver}}$, and the case of $\mathcal{O} = \mathcal{O}_{\mathsf{OSS.Sign}}$ follows essentially the same way (note the symmetry between A_y and $A_y^{\perp}$ in our proof).

Suppose for a contradiction that incompressibility is not satisfied: There exists query-bounded $\mathcal{P}_1, \mathcal{P}_2$ and a polynomial $p(\cdot)$ such that $p_{win} \geq \frac{1}{p(\lambda)}$, where we define

$$p_{win} = \Pr\left[\begin{array}{c|c} \forall i \in [s(\lambda)+1]\ \mathcal{O}_{\mathsf{Ver}}(vk_i, m_i, sig_i) \neq 0 \\ \wedge \\ (vk_i)_{i\in[s(\lambda)+1]} \text{ distinct} \end{array} : \begin{array}{c} (\mathcal{O}_{\mathsf{OSS.GenQKey}}, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}) \leftarrow \mathsf{OSS.GenOracles}(1^\lambda) \\ L \leftarrow \mathcal{P}_1^{(\mathcal{O}_{\mathsf{OSS.GenQKey}}, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}})}(1^\lambda) \\ (vk_i, m_i, sig_i)_{i\in[s(\lambda)+1]} \leftarrow \mathcal{P}_2^{\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}}(L) \end{array}\right]$$

where $s(\lambda) = 2q(\lambda)$ and $q(\lambda)$ is the number of queries to $\mathcal{O}_{\mathsf{OSS.GenQKey}}$ made by $\mathcal{P}_1$.

Now define a sequence of hybrids for some adversaries $\mathcal{A}_1, \mathcal{A}_2$.

Hyb_0**:** We define this hybrid as follows.

$\underline{\mathsf{Hyb}_0}$

1. Sample a random permutation π on $\{0,1\}^n$.
2. For each $y \in \{0,1\}^r$, sample a random full column-rank matrix $A_y \in \mathbb{F}_2^{k\times(n-r)}$ and a random vector b_y in $\mathbb{F}_2^k$.
3. Set $\mathcal{P}(x) = (y, A_y \cdot J(x) + b_y)$ where $y = H(x)$.
4. Set $\mathcal{P}^{-1}(y, v) = \pi^{-1}(y||z)$ if $v = A_y \cdot z + b_y$ for some $z \in \mathbb{F}_2^{n-r}$, otherwise $\perp$.
5. Set $\mathcal{D}(y, v) = 1$ if $v^T A_y = 0$ and $v \neq 0$, otherwise 0.
6. Set $\mathcal{D}_0(y, m, v) = 1$ if $v = A_y \cdot z + b_y$ for some $z \in \mathbb{F}_2^{n-r}$ and the first bit of v is m; otherwise 0.
7. Set $\mathcal{O}_{\mathsf{Ver}} = \mathcal{D}_0$, $\mathcal{O}_{\mathsf{Sign}} = \mathcal{D}$, $\mathcal{O}_{\mathsf{GenQKey}} = (\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}, \mathcal{D}_0)$.
8. Sample $L \leftarrow \mathcal{A}_1^{(\mathcal{O}_{\mathsf{OSS.GenQKey}}, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}})}(1^\lambda)$.
9. Sample $(vk_i, m_i, v_i)_{i\in[q(\lambda)+1]} \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}}(L, (b_y)_{y\in\{0,1\}^r})$.
10. Check that vk_i are all distinct, and that $v_i + b_{vk_i}$ starts with the bit m_i and that $v_i \in \mathsf{Colspan}(A_{vk_i}) \setminus \{0\}$ for all $i \in [q+1]$. If so, output 1, otherwise output 0.

We set $\mathcal{A}_1$ to $\mathcal{P}_1$ and $\mathcal{A}_2$ to be the following adversary. It first runs $\mathcal{A}_1$ to obtain $(vk_i, m_i, sig_i)_{i\in[s(\lambda)+1]}$, then it checks if there are at least $q+1$ values of $i \in [s(\lambda)+1]$ such that $sig_i \neq b_{vk_i}$. If so, let $I \subset [s(\lambda)+1]$ (with $|I| = q+1$) denote those values and $\mathcal{A}_2$ outputs $(vk_i, m_i, sig_i - b_{vk_i})_{i\in I}$. Otherwise, it outputs $\perp$.

Then, we claim that we have $\Pr[\mathsf{Hyb}_0 = 1] \geq \frac{1}{2p}$. To see this, first define the event E_1 to be the event in the definition of p_{win}. Define E_2 to be the event that for at least $q+1$ values of $i \in [s+1]$, we have $sig_i \neq b_{vk_i}$ (where the same sampling procedure is the same as in the definition

of p_{win}). Now, observe that we have $\Pr[E_1 \wedge E_2] + \Pr\left[E_1 \wedge \overline{E_2}\right] = p_{win} \geq \frac{1}{p}$. Now we claim that $\Pr\left[E_1 \wedge \overline{E_2}\right] \leq \mathsf{negl}(\lambda)$. Observe that once we prove this, $\Pr[\mathsf{Hyb}_0 = 1] \geq \frac{1}{2p}$ follows easily for $\mathcal{A}_1, \mathcal{A}_2$ we chose above.

Now we prove $\Pr\left[E_1 \wedge \overline{E_2}\right] \leq \mathsf{negl}(\lambda)$. Observe that once $E_1 \wedge \overline{E_2}$ occurs, it means that we have recovered the value b_{vk} as part of the output $\mathcal{P}_2$ for at least $q+1$ different vk values. Also note that each query to $\mathcal{O}_{\mathsf{OSS.Ver}}$ made by $\mathcal{P}_2$ can be simulated by making 2^{n-r} queries to an oracle that verifies equality for b_y values (i.e. the oracle that outputs 1 on b_y, y and 0 otherwise). Thus, $\mathcal{P}_1$ makes q queries to the oracle $\mathcal{O}(y) = b_y$ and $\mathcal{P}_2$ makes $\mathsf{poly}(\lambda) \cdot 2^{n-r}$ queries to the verification/equality check oracle for b_y values. Thus, by Theorem 15 and by the values of the parameters n, r, k, we get that (except with negligibly small probability) $\mathcal{P}_2$ cannot output b_{vk} values for $q+1$ different vk values. Hence, $\Pr\left[E_1 \wedge \overline{E_2}\right] \leq \mathsf{negl}(\lambda)$ as desired.

Hyb_1: Let $t(\lambda)$ denote the total number of queries in the experiment. Set $\ell(\lambda) = 1024 \cdot C_{\mathsf{SRD}} \cdot p(\lambda) \cdot (t(\lambda))^3$ where C_{SRD} is the constant from Theorem 8.

Hyb_1

1. Sample a random permutation π on $\{0,1\}^n$.
2. For each $y \in \{0,1\}^r$, sample a random full column-rank matrix $A_y \in \mathbb{F}_2^{k \times (n-r)}$ and a random vector b_y in $\mathbb{F}_2^k$.
3. Sample a random function $SRD : \{0,1\}^r \to [\ell]$.
4. Set $\mathcal{P}(x) = (y, A_{SRD(y)} \cdot J(x) + b_y)$ where $y = H(x)$.
5. Set $\mathcal{P}^{-1}(y, v) = \pi^{-1}(y||z)$ if $v = A_{SRD(y)} \cdot z + b_y$ for some $z \in \mathbb{F}_2^{n-r}$, otherwise $\perp$.
6. Set $\mathcal{D}(y, v) = 1$ if $v^T A_{SRD(y)} = 0$ and $v \neq 0$, otherwise 0.
7. Set $\mathcal{D}_0(y, m, v) = 1$ if $v = A_{SRD(y)} \cdot z + b_y$ for some $z \in \mathbb{F}_2^{n-r}$ and the first bit of v is m; otherwise 0.
8. Set $\mathcal{O}_{\mathsf{Ver}} = \mathcal{D}_0$, $\mathcal{O}_{\mathsf{Sign}} = \mathcal{D}$, $\mathcal{O}_{\mathsf{GenQKey}} = (\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}, \mathcal{D}_0)$.
9. Sample $L \leftarrow \mathcal{A}_1^{(\mathcal{O}_{\mathsf{OSS.GenQKey}}, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}})}(1^\lambda)$.
10. Sample $(vk_i, m_i, v_i)_{i \in [q(\lambda)+1]} \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}}(L, (b_y)_{y \in \{0,1\}^r})$.
11. Check that vk_i are all distinct, and that $v_i + b_{vk_i}$ starts with the bit m_i and that $v_i \in \mathsf{Colspan}(A_{SRD(vk_i)}) \setminus \{0\}$ for all $i \in [q+1]$. If so, output 1, otherwise output 0.

We have $\mathsf{Hyb}_0 \approx_{1/16p} \mathsf{Hyb}_1$ by Theorem 8, since the whole experiment can be implemented with oracle access to the oracle $\mathcal{O}_A(y) = A_y$ (where each $\mathcal{O}_A(y)$ is sampled independently from the full-column-rank matrix distribution as above)

Hyb_2: We define this hybrid as follows.

$\underline{\mathsf{Hyb}_2}$

1. Sample a random permutation π on $\{0,1\}^n$.
2. For each $y \in \{0,1\}^r$, sample a random full column-rank matrix $A_y \in \mathbb{F}_2^{k\times(n-r)}$ and a random vector b_y in $\mathbb{F}_2^k$.
3. For each $y \in \{0,1\}^r$, sample $A'_y \in \mathbb{F}_2^{k\times(n-\frac{r}{2})}$ and $A''_y \in \mathbb{F}_2^{k\times(n-\frac{r}{2})}$ so that $\mathsf{Colspan}(A'_y)$ is a random superspace of $\mathsf{Colspan}(A_y)$ of dimension $n-\frac{r}{2}$ and $\mathsf{Colspan}(A''_y)^\perp$ is a random superspace of $\mathsf{Colspan}(A_y)^\perp$ of dimension $\frac{3r}{2}$.
4. Sample a random function $SRD : \{0,1\}^r \to [\ell]$.
5. Set $\mathcal{P}(x) = (y, A_{SRD(y)} \cdot J(x) + b_y)$ where $y = H(x)$.
6. Set $\mathcal{P}^{-1}(y,v) = \pi^{-1}(y||z)$ if $v = A_{SRD(y)} \cdot z + b_y$ for some $z \in \mathbb{F}_2^{n-r}$, otherwise $\perp$.
7. Set $\mathcal{D}(y,v) = 1$ if $v^T A''_{SRD(y)} = 0$ and $v \neq 0$, otherwise 0.
8. Set $\mathcal{D}_0(y,m,v) = 1$ if $v = A'_{SRD(y)} \cdot z + b_y$ for some $z \in \mathbb{F}_2^{n-r}$ and the first bit of v is m; otherwise 0.
9. Set $\mathcal{O}_{\mathsf{Ver}} = \mathcal{D}_0$, $\mathcal{O}_{\mathsf{Sign}} = \mathcal{D}$, $\mathcal{O}_{\mathsf{GenQKey}} = (\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}, \mathcal{D}_0)$.
10. Sample $L \leftarrow \mathcal{A}_1^{(\mathcal{O}_{\mathsf{OSS.GenQKey}}, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}})}(1^\lambda)$.
11. Sample $(vk_i, m_i, v_i)_{i\in[q(\lambda)+1]} \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}}(L, (b_y)_{y\in\{0,1\}^r})$.
12. Check that vk_i are all distinct, and that $v_i + b_{vk_i}$ starts with the bit m_i and that $v_i \in \mathsf{Colspan}(A_{SRD(vk_i)}) \setminus \mathsf{Colspan}(A''_{SRD(vk_i)})$ for all $i \in [q+1]$. If so, output 1, otherwise output 0.

We have $\mathsf{Hyb}_1 \approx \mathsf{Hyb}_2$ by the following argument. First, note that the changes to the oracles $\mathcal{D}, \mathcal{D}_0$ make negligible difference by Lemma 2 (since we only deal with ℓ, which is polynomial, many subspaces). Also note that Lemma 2 allows the adversary to choose the base subspace, thus, the oracles $\mathcal{P}, \mathcal{P}^{-1}$ and so on do not preclude us from invoking the theorem. After we modify the oracles, the change to the winning condition also makes negligible difference by Theorem 14.

Hyb_3**:** We define this hybrid as follows.

$\underline{\mathsf{Hyb}_3}$

1. Sample a random permutation π on $\{0,1\}^n$.
2. For each $y \in \{0,1\}^r$, sample a random full column-rank matrix $A_y \in \mathbb{F}_2^{k\times(n-r)}$ and a random vector b_y in $\mathbb{F}_2^k$.
3. For each $y \in \{0,1\}^r$, sample $A'_y \in \mathbb{F}_2^{k\times(n-\frac{r}{2})}$ and $A''_y \in \mathbb{F}_2^{k\times(n-\frac{r}{2})}$ so that $\mathsf{Colspan}(A'_y)$ is a random superspace of $\mathsf{Colspan}(A_y)$ of dimension $n-\frac{r}{2}$ and $\mathsf{Colspan}(A''_y)^\perp$ is a random

superspace of $\mathsf{Colspan}(A_y)^\perp$ of dimension $\frac{3r}{2}$ (or, equivalently, $\mathsf{Colspan}(A''_y)$ is a random subspace of $\mathsf{Colspan}(A_y)$ of dimension $n - \frac{3r}{2}$.)

4. Set $\mathcal{P}(x) = (y, A_y \cdot J(x) + b_y)$ where $y = H(x)$.
5. Set $\mathcal{P}^{-1}(y, v) = \pi^{-1}(y||z)$ if $v = A_y \cdot z + b_y$ for some $z \in \mathbb{F}_2^{n-r}$, otherwise $\perp$.
6. Set $\mathcal{D}(y, v) = 1$ if $v^T A''_y = 0$ and $v \neq 0$, otherwise 0.
7. Set $\mathcal{D}_0(y, m, v) = 1$ if $v = A'_y \cdot z + b_y$ for some $z \in \mathbb{F}_2^{n-r}$ and the first bit of v is m; otherwise 0.
8. Set $\mathcal{O}_{\mathsf{Ver}} = \mathcal{D}_0$, $\mathcal{O}_{\mathsf{Sign}} = \mathcal{D}$, $\mathcal{O}_{\mathsf{GenQKey}} = (\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}, \mathcal{D}_0)$.
9. Sample $L \leftarrow \mathcal{A}_1^{(\mathcal{O}_{\mathsf{OSS.GenQKey}}, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}})}(1^\lambda)$.
10. Sample $(vk_i, m_i, v_i)_{i \in [q(\lambda)+1]} \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}}(L, (b_y)_{y \in \{0,1\}^r})$.
11. Check that vk_i are all distinct, and that $v_i + b_{vk_i}$ starts with the bit m_i and that $v_i \in \mathsf{Colspan}(A_{vk_i}) \setminus \mathsf{Colspan}(A''_{vk_i})$ for all $i \in [q+1]$. If so, output 1, otherwise output 0.

We have $\mathsf{Hyb}_2 \approx_{1/16p} \mathsf{Hyb}_3$ by Theorem 8, since the whole experiment can be implemented with oracle access to the oracle $\mathcal{O}_A(y) = A''_y||A_y||A'_y$ (where each $\mathcal{O}_A(y)$ is sampled independently as in the experiment).

Hyb_{final}: We now define our final hybrid, for some adversaries $\mathcal{A}'_1, \mathcal{A}'_2$.

Hyb_{final}

1. Sample a random permutation π on $\{0,1\}^n$.
2. For each $y \in \{0,1\}^r$, sample a random vector b_y in $\mathbb{F}_2^k$.
3. For each $y \in \{0,1\}^r$, sample $A'_y \in \mathbb{F}_2^{k \times (n - \frac{r}{2})}$ and $A''_y \in \mathbb{F}_2^{k \times (n - \frac{r}{2})}$ so that $\mathsf{Colspan}(A'_y)$ is a random subspace of $\mathbb{F}_2^k$ of dimension $n - \frac{r}{2}$ and so that $\mathsf{Colspan}(A''_y)$ is a random subspace of $\mathsf{Colspan}(A'_y)$ of dimension $n - \frac{3r}{2}$.
4. For each $y \in \{0,1\}^r$, sample $A_y \in \mathbb{F}_2^{k \times (n-r)}$ so that $\mathsf{Colspan}(A_y)$ is a random subspace from the set $\mathcal{S}$ of subspaces of dimension $n - r$ where each subspace in the set is a superspace of $\mathsf{Colspan}(A''_y)$ and subspace of $\mathsf{Colspan}(A'_y)$.
5. Define the oracle $\mathcal{O}(y)$ that outputs the index of $\mathsf{Colspan}(A_y)$ in the set $\mathcal{S}$ when it is ordered in a canonical way. That is, each $\mathcal{O}(y)$ is independently and uniformly sampled from $[|\mathcal{S}|]$ (we will refer to the elements of $[|\mathcal{S}|]$ and the subspaces A_y in an interchangeable way).
6. Sample $L \leftarrow \mathcal{A}_1^{\mathcal{O}}(1^\lambda, \pi, (b_y, A'_y, A''_y)_{y \in \{0,1\}^r})$.
7. Sample $(vk_i, m_i, v_i)_{i \in [q(\lambda)+1]} \leftarrow \mathcal{A}_2(L, \pi, (b_y, A'_y, A''_y)_{y \in \{0,1\}^r})$.

8. **Winning Condition Check:** Check that vk_i are all distinct, and that $v_i + b_{vk_i}$ starts with the bit m_i and that $v_i \in \mathsf{Colspan}(A_{vk_i}) \setminus \mathsf{Colspan}(A''_{vk_i})$ for all $i \in [q+1]$. If so, output 1, otherwise output 0.

Since $\Pr[\mathsf{Hyb}_3 = 1] \geq \frac{1}{2p}$, it is easy to see that there exists $\mathcal{A}'_1, \mathcal{A}'_2$ such that $\mathcal{A}'_1$ makes q queries to $\mathcal{O}$ and $\Pr\left[\mathsf{Hyb}_{final} = 1\right] \geq \frac{1}{2p}$. Now we will show that this is a contradiction. Moving forward, without loss of generality, we will assume that $\mathcal{A}_2$ never outputs (i) $v_i \notin \mathsf{Colspan}(A'_{vk_i}) \setminus \mathsf{Colspan}(A''_{vk_i})$, (ii) it never outputs non-distinct vk_i values and (iii) it never outputs $\perp$ (e.g. at worst, it can output random vectors in $\mathsf{Colspan}(A'_{vk_i}) \setminus \mathsf{Colspan}(A''_{vk_i})$). Now, first, consider the compressed oracle simulation, Exp of the hybrid Hyb_{final}. In particular, we also execute the final part of the experiment (i.e. winning condition check) using queries to $\mathcal{O}$: For $i \in [q+1]$, we first query the oracle $\mathcal{O}$ on vk_i and write the output to an empty register, and then we perform the membership check for v_i by reading from this register. We assume that the output (win or lose bit) is also written to a new register (call it the experiment outcome register). Since compressed oracle simulation is a perfect simulation, we have $\Pr[\mathsf{Exp} = 1] = \Pr\left[\mathsf{Hyb}_{final} = 1\right]$. Now we define the experiment Exp' where we modify the execution of the compressed oracle simulation during the winning condition check as follows. Instead of applying $\mathsf{StdDecomp} \circ \mathsf{CStO}' \circ \mathsf{StdDecomp}$ (we assume that $\mathsf{Increase}$ has been applied for sufficiently many times in advance), we instead apply $\mathsf{StdDecomp} \circ \mathsf{CStO}' \circ \mathsf{StdDecomp}'$ where $\mathsf{StdDecomp}'$ is a linear mapping defined as $\mathsf{StdDecomp}' : |vk\rangle |v\rangle |D\rangle \to |vk\rangle |v\rangle \sum_{A \in [|S|] : v \notin \mathsf{Colspan}(A) \setminus \mathsf{Colspan}(A''_{vk})} \frac{1}{\sqrt{a}} |D \cup (vk, A)\rangle$ for D, vk such that $D(vk) = \perp$ (and defined the same as $\mathsf{StdDecomp}$ for other basis elements), where $a = \begin{bmatrix} r \\ r/2 \end{bmatrix}_2 - \begin{bmatrix} r-1 \\ r/2-1 \end{bmatrix}_2$. We will call the case of D, vk such that $D(vk) = \perp$ a *good case*. Since the original mapping $\mathsf{StdDecomp}$ maps $|vk\rangle |v\rangle |D\rangle \to |vk\rangle |v\rangle \sum_{A \in [|S|]} \frac{1}{\sqrt{|S|}} |D \cup (vk, A)\rangle$ for D such that $D(vk) = \perp$, and since $|S| = \begin{bmatrix} r \\ r/2 \end{bmatrix}_2$, it is easy to see that $\mathsf{Exp}' \approx \mathsf{Exp}$. Finally, observe that $\Pr\left[\mathsf{Exp}' = 1\right] = 0$: For any output $(vk_i, m_i, v_i)_{i \in [q(\lambda)+1]}$ (of $\mathcal{A}_2$) and any compressed oracle database D that is in the support of the system when $\mathcal{A}_2$ yields its output, it will be the case that there is at least one value $i \in [q+1]$ such that $vk_i \notin D$ since D contains q valid pairs (also recall that when the good case does not apply, the number of valid pairs in the database does not increase). Thus, while applying the winning condition check, there will be a point where the good case of $\mathsf{StdDecomp}'$ applies, and after that it is easy to see that verification for that index fails. Thus, any $(vk_i, m_i, v_i)_{i \in [q(\lambda)+1]}$ and any compressed oracle database D in the support of the system will map to a state where the experiment outcome register will have the state $|0\rangle$. Thus, $\Pr\left[\mathsf{Exp}' = 1\right] = 0$. However, we had $\mathsf{Exp}' \approx \mathsf{Exp}$ and $\Pr[\mathsf{Exp} = 1] = \Pr\left[\mathsf{Hyb}_{final} = 1\right]$ and $\Pr\left[\mathsf{Hyb}_{final} = 1\right] \geq \frac{1}{2p}$. Thus, this is a contradiction, as desired.

7 Incompressibility of Random Oracles

In this section, we show that random oracles satisfy a natural notion of incompressibility. First we give our theorem statement regarding this property, then we prove a technical lemma that will help us in our proof of the incompressibility theorem, and finally we prove our incompressibility theorem.

Theorem 15 (Incompressibility of Random Oracles)**.** *Let $H : \{0,1\}^{g_1(\lambda)} \to \{0,1\}^{g_2(\lambda)}$ be a random oracle and define $\mathcal{O}_{\mathsf{ImageVer}}$ to be the following oracle (called* (image) verification oracle*).*

$\underline{\mathcal{O}_{\mathsf{ImageVer}}(x,y)}$
Hardcoded: H

1. *Check if $y = H(x)$. If so, output 1. Otherwise, output 0.*

For a predicate $P : \{0,1\}^{g_1(\lambda)} \to \{0,1\}$, define $\mathcal{O}_P$ as follows.

$\underline{\mathcal{O}(x)}$
Hardcoded: H, P

1. *Check if $P(x) = 0$. If so, output $\perp$ and terminate.*
2. *Output $H(x)$.*

Define the following experiment Exp_H.

$\underline{\mathsf{Exp}_H}$

1. *Set $i = 1$.*
2. *For any polynomial number of rounds (denote this by $u(\lambda)$):*
   1. *$\mathcal{A}_1$ makes polynomial number of queries (denote the number by $s_i(\lambda)$) to H and unbounded number of queries to $\mathcal{O}_{\cup_{j=1}^{i-1} P_j}$ where $\cup_{j=1}^{i-1} P_j$ denotes the union of the predicates $P_1, \ldots, P_{i-1}$ (i.e. $\cup_{j=1}^{i-1} P_j(x) = 1$ if $P_j(x) = 1$ for some $j = 1, \ldots, i-1$).*
   2. *$\mathcal{A}_1$ outputs a predicate P_i.*
   3. *Increase i by one.*
3. *$\mathcal{A}$ outputs an unbounded length message L.*

For any $g_1(\lambda), g_2(\lambda)$ such that $g_2(\lambda)$ is superlogarithmic, for an adversary $\mathcal{A}_1$ as described above, there exists a mapping $\mathcal{M}$ (which outputs a list[22] $\mathfrak{L}$ of size at most $\sum_{j \in [u(\lambda)]} s_j(\lambda)$) such that for all $\mathcal{A}_2$ that makes at most $q(\lambda)$ queries to $\mathcal{O}_{\mathsf{ImageVer}}$ such that $\frac{q(\lambda)}{\sqrt{2^{g_2(\lambda)}}}$ is negligible (but can make any number of queries to $\mathcal{O}_{P^*}$), we have

[22] We will refer to this as the incompressibility list.

$$\mathbb{E}\left[\sum_{\substack{x,y:\\ \mathcal{O}_{\mathsf{ImageVer}}(x,y)\neq 0\\ x\notin\mathcal{M}(H,L,(P_j)_j,\mathcal{A}_2)\\ P^*(x)=0}} \mathsf{QW}^{x,y}_{\mathcal{A}_2,L,H,(P_j)_j} : \begin{array}{c} H\leftarrow\{0,1\}^{g_1(\lambda)}\to\{0,1\}^{g_2(\lambda)}\\ L,(P_j)_{j\in[u(\lambda)]}\leftarrow\mathcal{A}_1^{H(\cdot)}(1^\lambda)\end{array}\right]\leq \mathsf{negl}(\lambda).$$

where $P^*=\cup_{j\in[u(\lambda)]}P_j$ *and* $\mathsf{QW}^{x,y}_{\mathcal{A}_2,L,H,(P_j)_j}$ *denotes the total query weight of* $\mathcal{A}_2^{\mathcal{O}_{P^*},\mathcal{O}_{\mathsf{ImageVer}}}(L)$ *to the oracle* $\mathcal{O}_{\mathsf{ImageVer}}$ *on oracle input* x,y*.*

We will refer to the queries to $\mathcal{O}_{P^*}$ (for $\mathcal{A}_2$) and $\mathcal{O}_{\cup_{j=1}^{i-1}P_j}$ (for $\mathcal{A}_1$) as *section queries* (and to these oracles as *section oracles*), and queries to $\mathcal{O}_{\mathsf{ImageVer}}$ as verification queries. A normal query will mean a query to H.

Before proving our theorem, we will prove a different technical lemma that will be useful in our proof. This lemma roughly says that, for a compressed random oracle adversary with additional verification oracle access to the random oracle; if at any point, the weight of databases that contain at most i inputs that satisfy a certain predicate is negligible, then this still holds after a verification query.

Lemma 3. *Let* $\mathcal{A}$ *be an adversary with access to a random oracle* $H:\{0,1\}^{m(\lambda)}\to\{0,1\}^{n(\lambda)}$ *and the verification oracle for the random oracle that outputs* 1 *on input* x,y *if* $H(x)=y$*, and outputs* 0 *otherwise. Define the registers*

- X,Y*, the input registers to the verification oracle,*
- Q*, the 1-qubit phase register for the verification oracle output,*
- W*, the internal workspace register of the adversary,*
- P*, the register containing a predicate*[23]*, chosen by the adversary,*
- D *compressed oracle database register,*
- T*, the register used for implementing the verification oracle queries (initialized at the beginning to zeroes and is not accessed by the adversary).*

Consider the following simulation of execution of $\mathcal{A}$*:*

- *The random oracle queries are implemented as a standard oracle (i.e. the output XORed into the output register) using a compressed oracle simulation,*
- *The verification oracle queries are implemented as a phase oracle as follows:*
  1. *Query the random oracle with input and output registers* X *and the output register* T*.*
  2. *Apply a controlled phase gate that maps* $|x\rangle|y\rangle|q\rangle|t\rangle$ *to* $(-1)^{b\cdot q}|x\rangle|y\rangle|q\rangle|t\rangle$ *where* $b=1$ *if and only if* $y=t$ *(and otherwise* $b=0$*).*
  3. *Query the random oracle with input and output registers* X *and the output register* T*.*

Then,

[23] We will write P to both mean the predicate function, and to mean the set of inputs x such that $P(x)=1$. Similarly, $\overline{P}$ means the complement of the set P

- *This is a perfect simulation of execution of $\mathcal{A}$,*
- *Let $|\phi\rangle = \sum_{x,y,q,w,P,D} \alpha_{x,y,q,w,P,D} |x\rangle |y\rangle |q\rangle |w\rangle |P\rangle |D\rangle \left|0^{n(\lambda)}\right\rangle$ be the state of the system right before executing one of the verification oracle queries, respectively over the registers $\mathsf{X}, \mathsf{Y}, \mathsf{Q}, \mathsf{W}, \mathsf{P}, \mathsf{D}, \mathsf{T}$. Let $i \in \mathbb{N}$ and let $\mathsf{Inp}(D)$ denote the set consisting of all a such that $(a, b_a) \in D$ for some $b_a \neq \perp$. Let let $|\psi\rangle = \sum_{x,y,q,w,P,D} \beta_{x,y,q,w,P,D} |x\rangle |y\rangle |q\rangle |w\rangle |P\rangle |D\rangle \left|0^{n(\lambda)}\right\rangle$ be the state right after executing the verification oracle query on $|\phi\rangle$.*

  Then, $\sum_{\substack{x,y,q,w,P,D:\\ |\mathsf{Inp}(D)\cap\overline{P}|>i}} |\beta_{x,y,q,w,P,D}|^2 \leq \sum_{\substack{x,y,q,w,P,D:\\ |\mathsf{Inp}(D)\cap\overline{P}|>i}} |\alpha_{x,y,q,w,P,D}|^2 + \frac{6}{\sqrt{2^{n(\lambda)}}}$.

Proof. The fact this simulation is a perfect simulation follows easily from the fact that compressed oracle simulation is a perfect simulation for the random oracle ([Zha19]). Now we prove the second part. Let CtPh denote the unitary for the controlled phase described above that is used in the implementation of the verification queries. Let $U = \mathsf{CStO} \circ \mathsf{CtPh} \circ \mathsf{CStO}$ (where CStO is the compressed oracle query unitary with input register X and output register T). Note that U is an isometry. Thus, $|\psi\rangle = U|\phi\rangle$ (here our notation means applying the isometry U to the state $|\phi\rangle$). We also let π denote[24] the projection onto $|P\rangle|D\rangle$ such that $|\mathsf{Inp}(D) \cap \overline{P}| > i$. Thus, we have $\sum_{\substack{x,y,q,w,P,D:\\ |\mathsf{Inp}(D)\cap\overline{P}|>i}} |\beta_{x,y,q,w,P,D}|^2 = \|\pi U|\phi\rangle\|^2$.

We write

$$|\phi\rangle = \sum_{\substack{x,y,q,w,P,D:\\ |\mathsf{Inp}(D)\cap\overline{P}|>i}} \alpha_{x,y,q,w,P,D} |x\rangle |y\rangle |q\rangle |w\rangle |P\rangle |D\rangle \left|0^{n(\lambda)}\right\rangle + \sum_{\substack{x,y,q,w,P,D:\\ |\mathsf{Inp}(D)\cap\overline{P}|\leq i\\ x\in D}} \alpha_{x,y,w,P,D} |x\rangle |y\rangle |q\rangle |w\rangle |P\rangle |D\rangle \left|0^{n(\lambda)}\right\rangle +$$
$$\sum_{\substack{x,y,q,w,P,D:\\ |\mathsf{Inp}(D)\cap\overline{P}|\leq i\\ x\notin D\\ P(x)=1}} \alpha_{x,y,w,P,D} |x\rangle |y\rangle |q\rangle |w\rangle |P\rangle |D\rangle \left|0^{n(\lambda)}\right\rangle + \sum_{\substack{x,y,q,w,P,D:\\ |\mathsf{Inp}(D)\cap\overline{P}|\leq i\\ x\notin D\\ P(x)=0}} \alpha_{x,y,q,w,P,D} |x\rangle |y\rangle |q\rangle |w\rangle |P\rangle |D\rangle \left|0^{n(\lambda)}\right\rangle$$

Let $|\phi_1\rangle, |\phi_2\rangle, |\phi_3\rangle, |\phi_4\rangle$ denote each unnormalized state in the summation above in the order they are written. Observe that when executing a verification query on $|x\rangle|y\rangle|q\rangle|w\rangle|P\rangle|D\rangle$, only x can be added to the database (if not already in the database), and the other entries in the database do not change. Thus, we get $\pi U|\phi_2\rangle = 0$. By the same argument, we get $\pi U|\phi_3\rangle = 0$. We also have $\|\pi U|\phi_1\rangle\|^2 \leq \||\phi_1\rangle\|^2 = \sum_{\substack{x,y,q,w,P,D:\\ |\mathsf{Inp}(D)\cap\overline{P}|>i}} |\alpha_{x,y,q,w,P,D}|^2$.

Now we claim $\|\pi U|\phi_4\rangle\|^2 \leq \frac{4}{2^n}$ will prove our claim. Observe that once we prove this, the theorem follows. To see this, first observe that $\sum_{\substack{x,y,q,w,P,D:\\ |\mathsf{Inp}(D)\cap\overline{P}|>i}} |\beta_{x,y,q,w,P,D}|^2 = \|\pi U|\phi\rangle\|^2 \leq \|\pi U|\phi_1\rangle\|^2 + \|\pi U|\phi_4\rangle\|^2 + 2\cdot\|\pi U|\phi_1\rangle\|\cdot\|\pi U|\phi_4\rangle\|$ by triangle inequality and since $\pi U|\phi_2\rangle = \pi U|\phi_3\rangle = 0$. Finally, note that $\|\pi U|\phi_1\rangle\|^2 + \|\pi U|\phi_4\rangle\|^2 + 2\cdot\|\pi U|\phi_1\rangle\|\cdot\|\pi U|\phi_4\rangle\| = \|\pi U|\phi_1\rangle\|^2 + \|\pi U|\phi_4\rangle\|(\|\pi U|\phi_4\rangle\| + 2\|\pi U|\phi_1\rangle\|)$ and $(\|\pi U|\phi_4\rangle\| + 2\|\pi U|\phi_1\rangle\|) \leq 3$.

Now we move onto proving $\|\pi U|\phi_4\rangle\|^2 \leq \frac{4}{2^n}$. For all valid states (i.e. states where compressed oracle invariants are kept) $|\tau\rangle$, we have $U|\tau\rangle = \mathsf{StdDecomp} \circ \mathsf{CStO}' \circ \mathsf{CPh} \circ \mathsf{CStO}' \circ \mathsf{StdDecomp} \circ (\mathsf{Inc})^2 |\tau\rangle$. This follows from the fact that unitaries on different registers commute, that $\mathsf{Increase}$ can be moved to the beginning, and that $\mathsf{StdDecomp}$ is an involution. Write $U_1 = \mathsf{CStO}' \circ \mathsf{StdDecomp} \circ (\mathsf{Inc})^2$ and $U_3 = \mathsf{StdDecomp} \circ \mathsf{CStO}'$; and thus $U = U_3 \circ \mathsf{CPh} \circ U_1$.

[24] By abuse of notation, we will also write π to denote the measurement $\pi \otimes I$ where the identity I is over the remaining registers.

After applying U_1 to $|\phi_4\rangle$, the state becomes

$$\sum_{\substack{x,y,q,w,P,D,z:\\ z\in\{0,1\}^{n(\lambda)}\\ |\mathsf{Inp}(D)\cap\overline{P}|\leq i\\ x\notin D\\ P(x)=0}} \alpha_{x,y,q,w,P,D}\frac{1}{\sqrt{2^{n(\lambda)}}}|x\rangle|y\rangle|q\rangle|w\rangle|P\rangle|D\cup(x,z)\cup(\bot,0^n)\rangle|z\rangle$$

Once we apply the controlled phase based on the verification, the state becomes

$$\sum_{\substack{x,y,q,w,P,D:\\ |\mathsf{Inp}(D)\cap\overline{P}|\leq i\\ x\notin D\\ P(x)=0}} (-1)^q\alpha_{x,y,q,w,P,D}\frac{1}{\sqrt{2^{n(\lambda)}}}|x\rangle|y\rangle|q\rangle|w\rangle|P\rangle|D\cup(x,y)\cup(\bot,0^n)\rangle|y\rangle+$$

$$\sum_{\substack{x,y,q,w,P,D:\\ |\mathsf{Inp}(D)\cap\overline{P}|\leq i\\ x\notin D\\ P(x)=0}}\sum_{z\neq y}\alpha_{x,y,q,w,P,D}\frac{1}{\sqrt{2^{n(\lambda)}}}|x\rangle|y\rangle|q\rangle|w\rangle|P\rangle|D\cup(x,z)\cup(\bot,0^n)\rangle|z\rangle$$

which is equal to

$$\sum_{\substack{x,y,q,w,P,D:\\ |\mathsf{Inp}(D)\cap\overline{P}|\leq i\\ x\notin D\\ P(x)=0}} ((-1)^q-1)\alpha_{x,y,q,w,P,D}\frac{1}{\sqrt{2^{n(\lambda)}}}|x\rangle|y\rangle|q\rangle|w\rangle|P\rangle|D\cup(x,y)\cup(\bot,0^n)\rangle|y\rangle+$$

$$\sum_{\substack{x,y,q,w,P,D:\\ |\mathsf{Inp}(D)\cap\overline{P}|\leq i\\ x\notin D\\ P(x)=0}}\sum_{z}\alpha_{x,y,q,w,P,D}\frac{1}{\sqrt{2^{n(\lambda)}}}|x\rangle|y\rangle|q\rangle|w\rangle|P\rangle|D\cup(x,z)\cup(\bot,0^n)\rangle|z\rangle$$

Denote the two unnormalized states in the summation as $|\eta_1\rangle, |\eta_2\rangle$ respectively. Since $\sum_{x,y,q,w,P,D}|\alpha_{x,y,q,w,P,D}|^2=1$, we get $\||\eta_1\rangle\|^2\leq\frac{4}{2^n}$ and thus $\|\pi\circ U_3|\eta_1\rangle\|^2\leq\frac{4}{2^n}$. Now, we show $\pi\circ U_3|\eta_2\rangle=0$. Observe that once we apply U_3 to $|\eta_2\rangle$, we get

$$\sum_{\substack{x,y,w,P,D:\\ |\mathsf{Inp}(D)\cap\overline{P}|\leq i\\ x\notin D\\ P(x)=0}} \alpha_{x,y,q,w,P,D}|x\rangle|y\rangle|q\rangle|w\rangle|P\rangle|D\cup\{(\bot,0^n),(\bot,0^n)\}\rangle\left|0^{n(\lambda)}\right\rangle$$

Hence, $\pi\circ U_3|\eta_2\rangle=0$, which completes the proof. □

Now we prove Theorem 15.

Proof. We will prove the operational version of the lemma (similar to Definition 22) and the actual incompressibility theorem directly follows (as in Theorem 10).

We will consider the compressed oracle simulation of the operational game, as in Lemma 3. Note that this simulation is purified: The output of $\mathcal{A}_1$ is measured at the very end. Let $\mathsf{P}_1,\ldots,\mathsf{P}_{u(\lambda)}$ denote the registers on which the predicates will appear - without loss of generality, we can assume

that these registers exist from the beginning, and are initialized to zeroes (which will be interpreted as a predicate that rejects all inputs).

For any $j \in \mathbb{N}$, let π_j denote the projection onto database states (over the database register D) and predicate states (over the registers $\mathsf{P}_1, \ldots, \mathsf{P}_{u(\lambda)}$) such that $|\mathsf{Inp}(D) \cap \overline{(\cup_{j=1}^{u(\lambda)} P_j)}| > j$. Let $|\tau\rangle$ be the state right before some query to the verification oracle, or the random oracle, or the section oracle, or right before updating a predicate register; and let $|\tau'\rangle$ denote the state right after the execution of the query. We observe the following.

- If the query is a verification query, then $\|\pi_j |\tau'\rangle\|^2 \leq \|\pi_j |\tau\rangle\|^2 + \frac{6}{\sqrt{2^{g_2(\lambda)}}}$. This follows directly from Lemma 3.
- If the query is a section query, then $\|\pi_j |\tau'\rangle\|^2 \leq \|\pi_j |\tau\rangle\|^2$. This follows since when a compressed oracle query is executed on the state $|x\rangle |D\rangle$, only x can be added to the database, no other values.
- If the query is a normal query, then $\|\pi_{j+1} |\tau'\rangle\|^2 \leq \|\pi_j |\tau\rangle\|^2$. This is because a single query can add at most one input to the database, and cannot change the other entries (other than deleting an entry).
- If we are considering a point where a predicate register is being updated, then $\|\pi_j |\tau'\rangle\|^2 \leq \|\pi_j |\tau\rangle\|^2$. This is because we are considering the union of the predicates in the definition of π_j, and adding more predicates makes the *good* set $(\mathsf{Inp}(D) \cap \overline{(\cup P_j)})$ smaller.

Applying this to each query of the adversaries $\mathcal{A}_1, \mathcal{A}_2$, it is easy to see that $\pi_{u(\lambda)} |\eta\rangle \leq \mathsf{negl}(\lambda)$ where $|\eta\rangle$ is the state of the system right before $\mathcal{A}_2$ produces its output in the operational incompressibility game, since there is a total of $u(\lambda)$ normal queries. By Theorem 7, this means that except with negligibly small probability, $\mathcal{A}_2$ can output at most $u(\lambda)$ different valid input - output pairs such that the input satisfies $P^*(x) = 0$, which completes the proof of operational incompressibility. □

Part III
Quantum-Key Fire

8 Quantum Key-Fire for Signing in the Classical Oracle Model

In this section, we define quantum key-fire for signing in the classical oracle model.

For simplicity, we work with an oracle model where the oracle is sampled by $\mathsf{QKeyFireSign.Setup}$ (i.e. there is an independent oracle for each instance) and output as the serial number. However, we note that, through elementary tricks, constructions in this model can also be easily converted to the standard classical oracle model where there is a single global classical oracle $\mathcal{O}$ (that is sampled from some distribution) that everyone (adversaries and the construction) has quantum-query access to, and $\mathsf{QKeyFireSign.Setup}^{\mathcal{O}}$ simply outputs a classical string as the verification key/serial number instead of outputting oracles. Again for simplicity, we consider the leakage phase as two adversaries communicating over a classical channel: note that this is equivalent to the leakage phase from the original game.

Definition 23 (Quantum Key-Fire for Signing in the Classical Oracle Model)**.** *A public-key quantum key-fire scheme for signing* $\mathsf{QKeyFireSign}$ *in the classical oracle model consists of the following efficient algorithms.*

- $\mathsf{QKeyFireSign.Setup}(1^\lambda)$*: Takes in a security parameter, outputs an oracle* $\mathcal{O}$ *as the verification key/serial number and a* key state $\mathsf{R_{flame}}$*.*
- $\mathsf{QKeyFireSign.Clone}^{\mathcal{O}}(\mathsf{R_{key}})$*: Takes in a key register* $\mathsf{R_{key}}$*, outputs a new register* $\mathsf{R_{clone}}$ *along with the input register* $\mathsf{R_{key}}$*.*
- $\mathsf{QKeyFireSign.Sign}^{\mathcal{O}}(\mathsf{R_{key}}, m)$ *: Takes in a key register and a message* m*, outputs a signature.*
- $\mathsf{QKeyFireSign.Ver}^{\mathcal{O}}(m, sig)$*: Takes in a message and an alleged signature, outputs* 1 *or* 0*.*

Evaluation Correctness: *For all* $m \in \mathcal{M}$*,*

$$\Pr\left[\mathsf{QKeyFireSign.Ver}^{\mathcal{O}^*}(m, sig) = 1 : \begin{array}{c} \mathcal{O}^*, \mathsf{R_{key}} \leftarrow \mathsf{QKeyFireSign.Setup}(1^\lambda) \\ sig \leftarrow \mathsf{QKeyFireSign.Sign}^{\mathcal{O}^*}(\mathsf{R_{key}}, m) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Strong cloning correctness: *This is defined the same as before: We require that output of* $\mathsf{QKeyFireSign.Setup}(1^\lambda)$ *is a pure state* $|\psi_{\mathcal{O}^*}\rangle$ *and that clone on input* $|\psi_{\mathcal{O}^*}\rangle$ *outputs two copies,* $|\psi_{\mathcal{O}^*}\rangle \otimes |\psi_{\mathcal{O}^*}\rangle$*.*

Definition 24 (LOCC Leakage-Resilience Security for Quantum Key-Fire for Signing in the Classical Oracle Model)**.** *Let* $\mathsf{QKeyFireSign}$ *be a quantum key-fire scheme with classical oracles. Consider the following game between an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *and a challenger.*

$\underline{\mathsf{LR-SIGN-GAME}_{\mathcal{A}}(1^\lambda)}$

1. *The challenger samples* $sn, \mathsf{R_{key}} \leftarrow \mathsf{QKeyFireSign.Setup}(1^\lambda)$ *and submits* $\mathsf{R_{key}}$ *to the adversary* $\mathcal{A}_1$*.*
2. *Parse* $\mathcal{O} = sn$*.*

3. ***Leakage Phase:*** *The adversaries $\mathcal{A}_1^{\mathcal{O}}$ and $\mathcal{A}_2^{\mathcal{O}}$ interact classically by exchanging classical messages for any number of rounds.*

4. *The challenger samples a random message $m^* \leftarrow \mathcal{M}$.*

5. *The adversary $\mathcal{A}_2^{\mathcal{O}}$ receives m^*, outputs sig^*.*

6. *The challenger checks if $\mathsf{QKeyFireSign.Ver}^{\mathcal{O}}(m^*, sig^*) = 1$. If so, it outputs 1, otherwise it outputs 0.*

We say that the quantum key-fire scheme QKeyFireSign *satisfies* LOCC leakage-resilience security *if for any QPT (or query bounded, in case of unconditional oracle-based schemes) adversary $\mathcal{A}$, we have*

$$\Pr\Big[\mathsf{LR-SIGN-GAME}_{\mathcal{A}}(1^\lambda) = 1\Big] \leq \mathsf{negl}(\lambda).$$

Note that this model also gives us the following conclusion. If the oracles of the scheme can be efficiently implemented (which is the case for our construction in Section 9, assuming one-way functions), our oracle security proof will also imply that our scheme is secure assuming virtual black-box obfuscation in the plain model and thus heuristically secure assuming indistinguishability obfuscation. Separately, as discussed, through elementary tricks our model also can be converted to the standard classical oracle model with unconditional security.

9 Construction

In this section, we give a quantum key-fire construction relative to a quantumly-accessible classical oracle (Definition 23) for signing $\nu(\lambda)$-bit messages (where $\nu(\lambda)$ is superlogarithmic), then prove its signing and cloning correctness. The security is proven in Section 10.

In Section 9.1, we describe our building block: incompressible one-shot signatures (OSS). Then, in Section 9.2, we give our construction.

Before moving onto the building block and construction, we introduce some notation.

Oracle Algorithms For a classical oracle $\mathcal{O}$, let $U_{\mathcal{O}}$ denote its *query execution unitary*, that is, the unitary that maps $|x\rangle|w\rangle|z\rangle$ to $|x\rangle|w\rangle|z \oplus \mathcal{O}(x)\rangle$ where the registers respectively are the query input, workspace and query output registers. For a quantum query algorithm $\mathcal{B}$, there is an associated *algorithm unitary* $U_{\mathcal{B}}$, and the execution of $\mathcal{B}$ with input R and oracle access to $\mathcal{O}$ is simply applying $(\mathcal{U}_{\mathcal{O}} \cdot U_{\mathcal{B}})^{p(\lambda)}$ to $(\mathsf{R}, \mathsf{R}_{\mathsf{aux}})$ (where $p(\lambda)$ is some polynomial and $\mathsf{R}_{\mathsf{aux}}$ is initialized to zeroes), and outputting (the whole or part of) the resulting register (after possibly measuring some parts of it in the computational basis)

9.1 Building Block: Incompressible One-Shot Signature Scheme

We will use an incompressible one-shot signature scheme OSS for 1-bit messages relative to a classical oracle (Definition 21) as a building block.

We make the following assumptions about OSS for presentation clarity, and all of these are without loss of generality.

Let $t_1(\cdot), t_2(\cdot), t_3(\cdot), t_4(\cdot)$ be some polynomials.

- Since OSS is an incompressible OSS, it has split oracles (Definition 20): write $(\mathcal{O}_{\mathsf{OSS.GenQKey}}, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}) = \mathcal{O}_{\mathsf{OSS}}$ to denote its oracles. Let $U_{\mathsf{OSS.GenQKey}}$ be the algorithm unitary and let $U_{\mathcal{O}_{\mathsf{OSS.GenQKey}}}$ be the query execution unitary for $\mathsf{OSS.GenQKey}$, and similarly define $U_{\mathsf{OSS.Sign}}$, $U_{\mathcal{O}_{\mathsf{OSS.Sign}}}$ for $\mathsf{OSS.Sign}$.
- $\mathsf{OSS.GenQKey}^{\mathcal{O}_{\mathsf{OSS.GenQKey}}}(1^\lambda)$: Define $U'_{\mathcal{O}_{\mathsf{OSS.GenQKey}},\mathsf{OSS.GenQKey}} = (U_{\mathcal{O}_{\mathsf{OSS.GenQKey}}} \cdot U_{\mathsf{OSS.GenQKey}})^{t_2(\lambda)}$. Then, this algorithm proceeds as follows:
  1. Introduce the auxiliary input register $\mathsf{R}_{\mathsf{inp-aux}}$ initialized to $\left|0^{t_4(\lambda)}\right\rangle$.
  2. Apply $U'_{\mathcal{O}_{\mathsf{OSS.GenQKey}},\mathsf{OSS.GenQKey}}$ to $\mathsf{R}_{\mathsf{inp-aux}}$ and parse the output into the registers: $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{key}}, \mathsf{R}_{\mathsf{out-aux}}$.
  3. Measure R_{vk} to obtain the verification key vk.
  4. Output $\mathsf{R}_{\mathsf{key}}$ as the signing key and vk as the verification key.
- $\mathsf{OSS.Sign}^{\mathcal{O}_{\mathsf{OSS.Sign}}}(vk, \mathsf{R}_{\mathsf{sk}}, m)$: Define $U'_{\mathcal{O}_{\mathsf{OSS.Sign}},\mathsf{OSS.Sign}} = (U_{\mathcal{O}_{\mathsf{OSS.Sign}}} \cdot U_{\mathsf{OSS.Sign}})^{t_1(\lambda)}$. Then, this algorithm proceeds as follows:
  1. Introduce the auxiliary input register $\mathsf{R}_{\mathsf{inp-aux}}$ initialized to $\left|0^{t_3(\lambda)}\right\rangle$.
  2. Introduce the message register $\mathsf{R}_{\mathsf{msg}}$ initialized to $|m\rangle$.
  3. Introduce the verification key register R_{vk} initialized to $|vk\rangle$.
  4. Apply $U'_{\mathcal{O}_{\mathsf{OSS.Sign}},\mathsf{OSS.Sign}}$ to $(\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{msg}}, \mathsf{R}_{\mathsf{sk}}, \mathsf{R}_{\mathsf{inp-aux}})$ and parse the output into the registers: R_{vk}, $\mathsf{R}_{\mathsf{msg}}$, $\mathsf{R}_{\mathsf{sig}}$, $\mathsf{R}_{\mathsf{out-aux}}$ (registers R_{vk}, $\mathsf{R}_{\mathsf{msg}}$ are unmodified).
  5. Measure $\mathsf{R}_{\mathsf{sig}}$ to obtain the signature sig.

6. Output sig.

- $\mathsf{OSS.Ver}^{\mathcal{O}_{\mathsf{OSS.Ver}}}(vk, m, sig)$: Applies the oracle $\mathcal{O}_{\mathsf{OSS.Ver}}$ to obtain the output $\mathcal{O}_{\mathsf{OSS.Ver}}(vk, m, sig)$. If the output is 1, the algorithm outputs 1, otherwise it outputs 0[25].

9.2 Construction

Let $\nu(\lambda)$ be any polynomially bounded function. We now give our public-key quantum key-fire construction for signing $\nu(\lambda)$-bit messages, with the classical oracle $\mathcal{O}_{\mathsf{QKeyFireSign}} = (\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_3, \mathcal{O}_4)$.

Let OSS be a one-shot signature scheme for 1-bit messages as described in Section 9.1. Let $p_1(\lambda)$ denote the length of verification keys of OSS, $p_2(\lambda)$ denote the input length of the oracle $\mathcal{O}_{\mathsf{OSS.GenQKey}}$.

$\underline{\mathsf{QKeyFireSign.Setup}(1^\lambda)}$

1. Sample random functions $H_0, H_1, H_{\mathsf{sig}}$ where $H_0 : \{0,1\}^{p_1(\lambda)+p_2(\lambda)} \to \{0,1\}^\lambda$, $H_1 : \{0,1\}^{p_1(\lambda)+p_2(\lambda)} \to \{0,1\}^\lambda$.
2. Sample random function $H_{\mathsf{sig}} : \{0,1\}^{\nu(\lambda)} \to \{0,1\}^\lambda$.
3. Sample $(\mathcal{O}_{\mathsf{OSS.GenQKey}}, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}) \leftarrow \mathsf{OSS.GenOracles}(1^\lambda)$.
4. Construct the oracle $\mathcal{O}_0$.

> $\underline{\mathcal{O}_0(ivk, isig, z)}$
> **Hardcoded:** $\mathcal{O}_{\mathsf{OSS.Ver}}, H_0$
>
> 1. Check if $\mathcal{O}_{\mathsf{OSS.Ver}}(ivk, 0, isig) = 1$. If not, output $\perp$ and terminate.
> 2. Output $H_0(ivk||z)$.

5. Construct the oracle $\mathcal{O}_1$.

> $\underline{\mathcal{O}_1(ivk, isig, z)}$
> **Hardcoded:** $\mathcal{O}_{\mathsf{OSS.Ver}}, H_1$
>
> 1. Check if $\mathcal{O}_{\mathsf{OSS.Ver}}(ivk, 1, isig) = 1$. If not, output $\perp$ and terminate.
> 2. Output $H_1(ivk||z)$.

6. Construct the oracle $\mathcal{O}_2$.

[25]The assumption about $\mathsf{OSS.Ver}$ is essentially that the signature verification procedure is classical and deterministic. It is easy to see that deterministic verification assumption is without loss of generality due to correctness with overwhelming probability. When we do not require the oracles to be efficiently implementable, classical verification assumption is also without loss of generality since any *classical input - classical output* quantum algorithm can be simulated by a classical oracle. However, in the efficient oracle case, the assumption about $\mathsf{OSS.Ver}$ might not be without loss of generality, but it is nevertheless true for our OSS construction in Section 6. The assumptions about $\mathsf{OSS.GenQKey}$ and $\mathsf{OSS.Sign}$ are indeed without loss of generality in all cases.

$\underline{\mathcal{O}_2(ivk, y_0, y_1, z)}$

Hardcoded: $H_0, H_1, \mathcal{O}_{\mathsf{OSS.GenQKey}}$

1. Check if $y_0 = H_0(ivk||z)$ and if $y_1 = H_1(ivk||z)$. If not, output $\perp$ and terminate.
2. Output $\mathcal{O}_{\mathsf{OSS.GenQKey}}(z)$.

7. Construct the oracle $\mathcal{O}_3$.

$\underline{\mathcal{O}_3(ivk, y_0, y_1, m)}$

Hardcoded: $H_0, H_1, H_{\mathsf{sig}}$

1. Check if $y_0 = H_0(ivk||m)$ and if $y_1 = H_1(ivk||m)$. If not, output $\perp$ and terminate.
2. Output $H_{\mathsf{sig}}(m)$.

8. Construct the oracle $\mathcal{O}_4$.

$\underline{\mathcal{O}_4(m, y)}$

Hardcoded: H_{sig}

1. Output 1 if $H_{\mathsf{sig}}(m) = y$, and output 0 otherwise.

9. Set $sn = (\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_3, \mathcal{O}_4)$.
10. Simulate the purified version of $\mathsf{OSS.GenQKey}^{\mathcal{O}_{\mathsf{OSS.GenQKey}}}(1^\lambda)$ and let $(\mathsf{R_{vk}}, \mathsf{R_{key}}, \mathsf{R_{out-aux}})$ be the output registers.
11. Output $sn, (\mathsf{R_{vk}}, \mathsf{R_{key}}, \mathsf{R_{out-aux}})$.

$\underline{\mathsf{QKeyFireSign.Sign}^{\mathcal{O}}(\mathsf{R_{flame}}, m)}$

1. Parse $(\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}, \mathcal{O}_3, \mathcal{O}_4) = \mathcal{O}$.
2. Parse $(\mathsf{R_{vk}}, \mathsf{R_{key}}, \mathsf{R_{out-aux}}) = \mathsf{R_{flame}}$.
3. Measure $\mathsf{R_{vk}}$ to obtain ivk.
4. Coherently run the following to obtain y_0 and then rewind (as in Lemma 1):
    1. Simulate $\mathsf{OSS.Sign}^{\mathcal{O}_{\mathsf{OSS.Sign}}}(ivk, \mathsf{R_{key}}, 0)$, let $isig$ be the output. Run $\mathcal{O}_0(ivk, isig, m)$ and let y_0 be the output.
5. Coherently run the following to obtain y_1 and then rewind (as in Lemma 1):
    1. Simulate $\mathsf{OSS.Sign}^{\mathcal{O}_{\mathsf{OSS.Sign}}}(ivk, \mathsf{R_{key}}, 0)$, let $isig$ be the output. Run $\mathcal{O}_1(ivk, isig, m)$ and let y_1 be the output.
6. Output $\mathcal{O}_3(ivk, y_0, y_1, m)$.

$\underline{\mathsf{QKeyFireSign.Ver}^{\mathcal{O}}(m, sig)}$

1. Parse $(\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_3, \mathcal{O}_4) = \mathcal{O}$.
2. Output $\mathcal{O}_4(m, sig)$.

$\underline{\mathsf{QKeyFireSign.Clone}^{\mathcal{O}}(\mathsf{R}_{\mathsf{flame}})}$

1. Parse $(\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_3, \mathcal{O}_4) = \mathcal{O}$.
2. Parse $(\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{key}}, \mathsf{R}_{\mathsf{out-aux},1}) = \mathsf{R}_{\mathsf{flame}}$.
3. Create the registers $\mathsf{R}_{\mathsf{msg}}, \mathsf{R}_{\mathsf{aux},2}, \mathsf{R}_{\mathsf{aux},3}, \mathsf{R}_{\mathsf{ora-out},4}, \mathsf{R}_{\mathsf{ora-out},5}$. Initialize $\mathsf{R}_{\mathsf{aux},2}$ and $\mathsf{R}_{\mathsf{aux},3}$ to $\left|0^{t_3(\lambda)}\right\rangle$ and $\left|0^{t_4(\lambda)}\right\rangle$. Initialize $\mathsf{R}_{\mathsf{ora-out},4}$ and $\mathsf{R}_{\mathsf{ora-out},5}$ to $\left|0^{\lambda}\right\rangle$. Initialize $\mathsf{R}_{\mathsf{msg}}$ to $|0\rangle$. At some steps, we will parse $\mathsf{R}_{\mathsf{aux},3}$ as $\mathsf{R}_{\mathsf{ora-inp},3}, \mathsf{R}_{\mathsf{ora-out},3}, \mathsf{R}'_{\mathsf{aux},3}$, as in the oracle query step of $\mathsf{OSS.GenQKey}$, where $U_{\mathcal{O}_{\mathsf{OSS.GenQKey}}}$ reads the oracle input input from $\mathsf{R}_{\mathsf{ora-inp},3}$ and writes the oracle output onto $\mathsf{R}_{\mathsf{ora-out},3}$.
4. For $t_2(\lambda)$ steps, execute the following.
    1. Apply $U_{\mathsf{OSS.GenQKey}}$ to $\mathsf{R}_{\mathsf{aux},3}$.
    2. Execute the following (which essentially simulates the oracle call to $\mathcal{O}_{\mathsf{OSS.GenQKey}}$):
        1. Apply $U'_{\mathcal{O}_{\mathsf{OSS.Sign}},\mathsf{OSS.Sign}}$ to R_{vk}, $\mathsf{R}_{\mathsf{msg}}$,$\mathsf{R}_{\mathsf{key}}$, $\mathsf{R}_{\mathsf{aux},2}$. Parse the output as R_{vk},$\mathsf{R}_{\mathsf{msg}}$,$\mathsf{R}_{\mathsf{sig}}$, $\mathsf{R}_{\mathsf{out-aux},2}$ as in $\mathsf{OSS.Sign}$.
        2. Apply $U_{\mathcal{O}_0}$ with input registers $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{sig}}, \mathsf{R}_{\mathsf{ora-inp},3}$ and the output register $\mathsf{R}_{\mathsf{ora-out},4}$.
        3. Apply $(U'_{\mathcal{O}_{\mathsf{OSS.Sign}},\mathsf{OSS.Sign}})^{-1}$ to $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{msg}}, \mathsf{R}_{\mathsf{sig}}, \mathsf{R}_{\mathsf{aux},2}$, parse the output as R_{vk}, $\mathsf{R}_{\mathsf{msg}}$, $\mathsf{R}_{\mathsf{key}}$, $\mathsf{R}_{\mathsf{aux},2}$.
        4. Set $\mathsf{R}_{\mathsf{msg}}$ to $|1\rangle$.
        5. Apply $U'_{\mathcal{O}_{\mathsf{OSS.Sign}},\mathsf{OSS.Sign}}$ to $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{msg}}, \mathsf{R}_{\mathsf{key}}, \mathsf{R}_{\mathsf{aux},2}$. Parse the output as R_{vk},$\mathsf{R}_{\mathsf{msg}}$, $\mathsf{R}_{\mathsf{sig}}$, $\mathsf{R}_{\mathsf{out-aux},2}$ as in $\mathsf{OSS.Sign}$.
        6. Apply $U_{\mathcal{O}_1}$ with input registers $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{sig}}, \mathsf{R}_{\mathsf{ora-inp},3}$ and the output register $\mathsf{R}_{\mathsf{ora-out},5}$.
        7. Apply $U_{\mathcal{O}_2}$ with input registers $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{ora-out},4}, \mathsf{R}_{\mathsf{ora-out},5}$ and the output register $\mathsf{R}_{\mathsf{ora-out},3}$.
        8. Apply $U_{\mathcal{O}_1}$ with input registers $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{sig}}, \mathsf{R}_{\mathsf{ora-inp},3}$ and the output register $\mathsf{R}_{\mathsf{ora-out},5}$.
        9. Apply $(U'_{\mathcal{O}_{\mathsf{OSS.Sign}},\mathsf{OSS.Sign}})^{-1}$ to , $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{msg}}, \mathsf{R}_{\mathsf{sig}}, \mathsf{R}_{\mathsf{aux},2}$, parse the output as R_{vk}, $\mathsf{R}_{\mathsf{msg}}$, $\mathsf{R}_{\mathsf{key}}$, $\mathsf{R}_{\mathsf{aux},2}$.
        10. Set $\mathsf{R}_{\mathsf{msg}}$ to $|0\rangle$.
        11. Apply $U'_{\mathcal{O}_{\mathsf{OSS.Sign}},\mathsf{OSS.Sign}}$ to $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{msg}}$,$\mathsf{R}_{\mathsf{key}}$, $\mathsf{R}_{\mathsf{aux},2}$.
        12. Apply $U_{\mathcal{O}_0}$ with input registers $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{sig}}, \mathsf{R}_{\mathsf{ora-inp},3}$ and the output register $\mathsf{R}_{\mathsf{ora-out},4}$.
        13. Apply $(U'_{\mathcal{O}_{\mathsf{OSS.Sign}},\mathsf{OSS.Sign}})^{-1}$ to $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{msg}}, \mathsf{R}_{\mathsf{sig}}, \mathsf{R}_{\mathsf{aux},2}$, parse the output as R_{vk}, $\mathsf{R}_{\mathsf{msg}}$, $\mathsf{R}_{\mathsf{key}}$, $\mathsf{R}_{\mathsf{aux},2}$.
5. Parse $\mathsf{R}_{\mathsf{aux},3}$ as $\mathsf{R}^*_{\mathsf{vk}}, \mathsf{R}^*_{\mathsf{key}}, \mathsf{R}^*_{\mathsf{out-aux}}$ as in $\mathsf{OSS.GenQKey}$.
6. Set $\mathsf{R}_{\mathsf{clone}} = (\mathsf{R}^*_{\mathsf{vk}}, \mathsf{R}^*_{\mathsf{key}}, \mathsf{R}^*_{\mathsf{out-aux}})$.
7. Output $\mathsf{R}_{\mathsf{flame}}, \mathsf{R}_{\mathsf{clone}}$.

Remark 1. *As described above, the signing algorithm* QKeyFireSign.Sign *of our scheme consumes the flame state given to it. Note that thanks to the strong clonability correctness of the scheme, this is not a issue since one can clone the flame state before signing a message. However, if one insists on not consuming/altering the flame state while signing, this is also possible: Using the gentle measurement technique (Lemma 1) it is also easy to see that the signing algorithm can be made so that it does not alter the state of the flame register (except up to negligible trace distance), since the output is pseudodeterministic (it is equal to* $H_{\mathsf{sig}}(m)$ *with overwhelming probability).*

Remark 2. *Aside from the conversion from quantum protection schemes for signing to quantum protection schemes for general unlearnable functionalities shown by the previous work, we can also slightly modify our construction above to achieve key-fire for unlearnable functionalities directly: Instead of* $\mathcal{O}_3, \mathcal{O}_4$*, we will just have a new* $\mathcal{O}_3$ *which outputs* $f^*(x)$ *after verifying* y_0, y_1*. The proof also follows the same, with the difference that at the final hybrid, instead of invoking Theorem 21, we simply invoke the unlearnability of the functionality.*

We claim that the construction satisfies correctness and security.

Theorem 16. QKeyFireSign *satisfies quantum key-fire signing correctness, strong cloning correctness (Definition 18). Let* $\nu(\lambda)$ *be superlogarithmic. Suppose* OSS *satisfies* $\mathcal{O}_{\mathsf{OSS.Ver}}$ *incompressibility and one-shot security, then* QKeyFireSign *satisfies unbounded (i.e. non-interactive) leakage-resilience for signing unconditionally. Suppose* OSS *satisfies strong incompressibility and one-shot security, then* QKeyFireSign *satisfies LOCC leakage-resilience for signing (Definition 24) unconditionally. Further, the oracles of* QKeyFireSign *can be implemented efficiently using a one-way function while preserving security (in which case, the security becomes computational rather than information-theoretic).*

We prove the signing correctness and cloning correctness in Sections 9.3 and 9.4, and we prove security in Section 10.

By implementing our construction using the incompressible one-shot signature we construct in Theorem 9, we get the following corollary.

Theorem 17. *Relative to a classical oracle, there exists a quantum key-fire scheme that satisfies LOCC leakage-resilience for signing functionality.*

Theorem 18. *Relative to a classical oracle, there exists a quantum key-fire scheme that satisfies LOCC leakage-resilience for any unlearnable functionality.*

9.3 Proof of Signing Correctness

In this section, we prove the signing correctness of our scheme.

For simplicity, we will assume that OSS satisfies perfect correctness, and we will show that $\mathsf{QKeyFireSign}$ satisfies perfect signing correctness. Through elementary lemmas, this implies that if OSS satisfies correctness with overwhelming probability, then $\mathsf{QKeyFireSign}$ also satisfies signing correctness with overwhelming probability.

We first observe that the flame states of our scheme $(sn, |\psi\rangle_{\mathsf{R_{vk}},\mathsf{R_{key}},\mathsf{R_{out-aux}}})$ output by $\mathsf{QKeyFireSign.Setup}(1^\lambda)$ are the same as the purified output of $\mathsf{OSS.GenQKey}$ (whose original version traces out $\mathsf{R_{out-aux}}$ and measures $\mathsf{R_{vk}}$, but $\mathsf{QKeyFireSign.Setup}$ does neither since it is purified). Then, $\mathsf{QKeyFireSign.Sign}$ first measures $\mathsf{R_{vk}}$ to obtain some value ivk and (effectively) traces out $\mathsf{R_{out-aux}}$. This collapses the state to the one-shot signature key with the serial number ivk. By signing correctness of OSS and definition of $\mathcal{O}_0$, we get $y_0 = H_0(ivk||m)$ with probability 1. Thus, by Lemma 1, the signing key is indeed not altered since we are rewinding. Then again by signing correctness of OSS and definition of $\mathcal{O}_1$, we get $y_1 = H_1(ivk||m)$ with probability 1. Hence, the output of the signing algorithm is $\mathcal{O}_3(ivk, H_0(ivk||m), H_1(ivk||m), m)$ which is equal to $H_{\mathsf{sig}}(m)$. Due to the definition of the verification algorithm, this completes the proof of signing correctness.

9.4 Proof of Strong Cloning Correctness

In this section, we prove the strong cloning correctness of our scheme.

First, we note that the flame states of our scheme are indeed pure states as required for strong cloning correctness. Observe that the state output by $\mathsf{QKeyFireSign.Setup}$ is simply $|\psi_{flame}\rangle = (U_{\mathcal{O}_{\mathsf{OSS.GenQKey}}} \cdot U_{\mathsf{OSS.GenQKey}})^{t_2(\lambda)} \left|0^{t_4(\lambda)}\right\rangle$. This is also the same as the purified output of $\mathsf{OSS.GenQKey}$ (whose not-purified version traces out $\mathsf{R_{out-aux}}$ and measures $\mathsf{R_{vk}}$, but $\mathsf{QKeyFireSign.Setup}$ does neither).

For simplicity, we will assume that OSS satisfies perfect correctness, and we will show that $\mathsf{QKeyFireSign}$ satisfies perfect strong cloning correctness. Through elementary lemmas, this implies that if OSS satisfies correctness with overwhelming probability, then $\mathsf{QKeyFireSign}$ also satisfies strong cloning correctness with overwhelming probability.

We first prove the following lemma.

Lemma 4. *During the loop in* $\mathsf{QKeyFireSign.Clone}$*, if the state over the registers* $\mathsf{R_{vk}}$*,* $\mathsf{R_{key}}$*,* $\mathsf{R_{out-aux,1}}$*,* $\mathsf{R_{msg}}$*,* $\mathsf{R_{aux,2}}$*,* $\mathsf{R_{aux,3}}$*,* $\mathsf{R_{ora-out,4}}$*,* $\mathsf{R_{ora-out,5}}$ *is*

$$|\psi_{flame}\rangle_{\mathsf{R_{vk}},\mathsf{R_{key}},\mathsf{R_{out-aux,1}}} \otimes |0\rangle_{\mathsf{R_{msg}}} \otimes \left|0^{p_2(\lambda)}\right\rangle_{\mathsf{R_{aux,2}}} \otimes |\zeta\rangle_{\mathsf{R_{aux,3}}} \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R_{ora-out,4}}} \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R_{ora-out,5}}}$$

at the beginning of an iteration, then the state at the end of that iteration is

$$|\psi_{flame}\rangle_{\mathsf{R_{vk}},\mathsf{R_{key}},\mathsf{R_{out-aux,1}}} \otimes |0\rangle_{\mathsf{R_{msg}}} \otimes \left|0^{p_2(\lambda)}\right\rangle_{\mathsf{R_{aux,2}}} \otimes (U_{\mathcal{O}_{\mathsf{OSS.GenQKey}}} \cdot U_{\mathsf{OSS.GenQKey}} |\zeta\rangle) \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R_{ora-out,4}}} \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R_{ora-out,5}}}.$$

Proof. We will show the result by writing the state of the registers after each step of the iteration. The actual order of the registers will stay the same throughout, however, when writing the state we will reorder some registers for convenience of notation.

We also introduce the following notation for conciseness.

- $|\psi_{flame}\rangle = \sum_{vk,w} \beta_{vk,w} |vk\rangle_{\mathsf{R_{vk}}} \otimes |\phi_{vk,w}\rangle_{\mathsf{R_{key}}} \otimes |w\rangle_{\mathsf{R_{out-aux}}}$,
- $U_1 = U_{\mathsf{OSS.GenQKey}}$,

- $U_2 = U_{\mathcal{O}_{\mathsf{OSS.GenQKey}}}$,
- $U = U_2 \cdot U_1$,
- $U_1 |\zeta\rangle = \sum_{z,o,w'} \gamma_{z,o,w'} |z\rangle_{\mathsf{R}_{\mathsf{ora-inp},3}} \otimes |o\rangle_{\mathsf{R}_{\mathsf{ora-out},3}} \otimes |w'\rangle_{\mathsf{R}'_{\mathsf{aux},3}}$.

Now we show the evolution of the state during the loop.

1. After applying $U_{\mathsf{OSS.GenQKey}}$ to $\mathsf{R}_{\mathsf{aux},3}$, we get

$$|\psi_{flame}\rangle_{\mathsf{R}_{\mathsf{vk}},\mathsf{R}_{\mathsf{key}},\mathsf{R}_{\mathsf{out-aux},1}} \otimes |0\rangle_{\mathsf{R}_{\mathsf{msg}}} \otimes \left|0^{p_2(\lambda)}\right\rangle_{\mathsf{R}_{\mathsf{aux},2}} \otimes (U_1 |\zeta\rangle)_{\mathsf{R}_{\mathsf{aux},3}} \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R}_{\mathsf{ora-out},4}} \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R}_{\mathsf{ora-out},5}}$$

2. After applying $U'_{\mathcal{O}_{\mathsf{OSS.Sign}},\mathsf{OSS.Sign}}$ to $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{key}}, \mathsf{R}_{\mathsf{msg}}, \mathsf{R}_{\mathsf{aux},2}$, we get

$$|0\rangle_{\mathsf{R}_{\mathsf{msg}}} \otimes \left(\sum_{vk,w} \beta_{vk,w} |vk\rangle_{\mathsf{R}_{\mathsf{vk}}} \otimes \sum_{sig} \alpha_{vk,w,sig} |sig\rangle\right) \otimes (U_1 |\zeta\rangle)_{\mathsf{R}_{\mathsf{aux},3}} \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R}_{\mathsf{ora-out},4}} \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R}_{\mathsf{ora-out},5}}$$

   where for each vk, w, sig with $\beta_{vk,w} \neq 0$ and $\alpha_{vk,w,sig} \neq 0$ we have that $\mathcal{O}^*_{\mathsf{OSS.Ver}}(vk, sig, 0) = 1$ by correctness of $\mathsf{OSS.Sign}$.

3. After applying $U_{\mathcal{O}_0}$ with input registers $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{sig}}, \mathsf{R}_{\mathsf{ora-inp},3}$ and the output register $\mathsf{R}_{\mathsf{ora-out},4}$, we get

$$|0\rangle_{\mathsf{R}_{\mathsf{msg}}} \otimes \Bigg(\sum_{\substack{vk,w \\ z,o,w'}} \beta_{vk,w} |vk\rangle_{\mathsf{R}_{\mathsf{vk}}} \otimes (\sum_{sig} \alpha_{vk,w,sig}) \gamma_{z,o,w'} |z\rangle_{\mathsf{R}_{\mathsf{ora-inp},3}} \otimes$$
$$|o\rangle_{\mathsf{R}_{\mathsf{ora-out},3}} \otimes |\mathcal{O}_0(vk, sig_{vk,w}, z)\rangle_{\mathsf{R}_{\mathsf{ora-out},4}} \otimes \left|w'\right\rangle_{\mathsf{R}'_{\mathsf{aux},3}} \Bigg) \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R}_{\mathsf{ora-out},5}}$$

   which is equal to

$$|0\rangle_{\mathsf{R}_{\mathsf{msg}}} \otimes$$
$$\Bigg(\sum_{vk,w} \beta_{vk,w} \Bigg(|vk\rangle_{\mathsf{R}_{\mathsf{vk}}} \otimes (\sum_{z,o,w'} \gamma_{z,o,w'} |z\rangle_{\mathsf{R}_{\mathsf{ora-inp},3}} |o\rangle_{\mathsf{R}_{\mathsf{ora-out},3}} |H_0(vk,z)\rangle_{\mathsf{R}_{\mathsf{ora-out},4}} \otimes \left|w'\right\rangle_{\mathsf{R}'_{\mathsf{aux},3}}) \Bigg) \otimes$$
$$|sig_{vk,w}\rangle \Bigg) \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R}_{\mathsf{ora-out},5}}$$

4. After applying $(U'_{\mathcal{O}_{\mathsf{OSS.Sign}},\mathsf{OSS.Sign}})^{-1}$ to $\mathsf{R}_{\mathsf{crs}}, \mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{msg}}, \mathsf{R}_{\mathsf{sig}}, \mathsf{R}_{\mathsf{aux},2}$, we get

$$|0\rangle_{\mathsf{R}_{\mathsf{msg}}} \otimes$$
$$\Bigg(\sum_{vk,w} \beta_{vk,w} \Bigg(|vk\rangle_{\mathsf{R}_{\mathsf{vk}}} \otimes (\sum_{z,o,w'} \gamma_{z,o,w'} |z\rangle_{\mathsf{R}_{\mathsf{ora-inp},3}} |o\rangle_{\mathsf{R}_{\mathsf{ora-out},3}} |H_0(vk,z)\rangle_{\mathsf{R}_{\mathsf{ora-out},4}} \otimes \left|w'\right\rangle_{\mathsf{R}'_{\mathsf{aux},3}}) \Bigg) \otimes$$
$$|\phi_{vk,w}\rangle_{\mathsf{R}_{\mathsf{key}}} \otimes |w\rangle_{\mathsf{R}_{\mathsf{out-aux}}} \otimes \left|0^{p_2(\lambda)}\right\rangle_{\mathsf{R}_{\mathsf{aux},2}} \Bigg) \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R}_{\mathsf{ora-out},5}}$$

5. After applying X to $\mathsf{R}_{\mathsf{msg}}$, this register changes from $|0\rangle$ to $|1\rangle$, and the others are unaffected.

6. After applying $U'_{\mathcal{O}_{\mathsf{OSS.Sign}},\mathsf{OSS.Sign}}$ to $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{key}}, \mathsf{R}_{\mathsf{msg}}, \mathsf{R}_{\mathsf{aux},2}$ we get

$$|1\rangle_{\mathsf{R}_{\mathsf{msg}}} \otimes \Bigg(\sum_{vk,w} \beta_{vk,w} \Bigg(|vk\rangle_{\mathsf{R}_{\mathsf{vk}}} \otimes \Big(\sum_{z,o,w'} \gamma_{z,o,w'} |z\rangle_{\mathsf{R}_{\mathsf{ora-inp},3}} |o\rangle_{\mathsf{R}_{\mathsf{ora-out},3}} |H_0(vk,z)\rangle_{\mathsf{R}_{\mathsf{ora-out},4}} \otimes \left|w'\right\rangle_{\mathsf{R}'_{\mathsf{aux},3}} \Big) \Bigg) \otimes \sum_{sig} \alpha'_{vk,w,sig} |sig\rangle \Bigg) \otimes \left|0^\lambda\right\rangle_{\mathsf{R}_{\mathsf{ora-out},5}}$$

where for each vk, w, sig with $\beta_{vk,w} \neq 0$ and $\alpha'_{vk,w,sig} \neq 0$ we have that $\mathcal{O}^*_{\mathsf{OSS.Ver}}(vk, sig, 1) = 1$ by correctness of $\mathsf{OSS.Sign}$.

7. After applying $U_{\mathcal{O}_1}$ with input registers $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{sig}}, \mathsf{R}_{\mathsf{ora-inp},3}$ and the output register $\mathsf{R}_{\mathsf{ora-out},5}$, we get

$$|1\rangle_{\mathsf{R}_{\mathsf{msg}}} \otimes \Bigg(\sum_{vk,w} \beta_{vk,w} \Bigg(|vk\rangle_{\mathsf{R}_{\mathsf{vk}}} \otimes \Big(\sum_{z,o,w'} \gamma_{z,o,w'} |z\rangle_{\mathsf{R}_{\mathsf{ora-inp},3}} |o\rangle_{\mathsf{R}_{\mathsf{ora-out},3}} |H_0(vk,z)\rangle_{\mathsf{R}_{\mathsf{ora-out},4}} |H_1(vk,z)\rangle_{\mathsf{R}_{\mathsf{ora-out},5}} \otimes \left|w'\right\rangle_{\mathsf{R}'_{\mathsf{aux},3}} \Big) \Bigg) \otimes \sum_{sig} \alpha'_{vk,w,sig} |sig\rangle \Bigg)$$

8. After applying $U_{\mathcal{O}_2}$ with input registers $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{ora-out},4}, \mathsf{R}_{\mathsf{ora-out},5}$ and the output register $\mathsf{R}_{\mathsf{ora-out},3}$, we get

$$|1\rangle_{\mathsf{R}_{\mathsf{msg}}} \otimes \Bigg(\sum_{vk,w} \beta_{vk,w} \Bigg(|vk\rangle_{\mathsf{R}_{\mathsf{vk}}} \otimes \Big(\sum_{z,o,w'} \gamma_{z,o,w'} |z\rangle_{\mathsf{R}_{\mathsf{ora-inp},3}} \left|o \oplus \mathcal{O}^*_{\mathsf{OSS.GenQKey}(z)}\right\rangle_{\mathsf{R}_{\mathsf{ora-out},3}} |H_0(vk,z)\rangle_{\mathsf{R}_{\mathsf{ora-out},4}} |H_1(vk,z)\rangle_{\mathsf{R}_{\mathsf{ora-out},5}} \otimes \left|w'\right\rangle_{\mathsf{R}'_{\mathsf{aux},3}} \Big) \Bigg) \otimes \sum_{sig} \alpha'_{vk,w,sig} |sig\rangle \Bigg)$$

9. After applying $U_{\mathcal{O}_1}$ with input registers $\mathsf{R}_{\mathsf{vk}}, \mathsf{R}_{\mathsf{sig}}, \mathsf{R}_{\mathsf{ora-inp},3}$ and the output register $\mathsf{R}_{\mathsf{ora-out},5}$, we get

$$|1\rangle_{\mathsf{R}_{\mathsf{msg}}} \otimes \Bigg(\sum_{vk,w} \beta_{vk,w} \Bigg(|vk\rangle_{\mathsf{R}_{\mathsf{vk}}} \otimes \Big(\sum_{z,o,w'} \gamma_{z,o,w'} |z\rangle_{\mathsf{R}_{\mathsf{ora-inp},3}} \left|o \oplus \mathcal{O}^*_{\mathsf{OSS.GenQKey}(z)}\right\rangle_{\mathsf{R}_{\mathsf{ora-out},3}} |H_0(vk,z)\rangle_{\mathsf{R}_{\mathsf{ora-out},4}} \left|0^\lambda\right\rangle_{\mathsf{R}_{\mathsf{ora-out},5}} \otimes \left|w'\right\rangle_{\mathsf{R}'_{\mathsf{aux},3}} \Big) \Bigg) \otimes \sum_{sig} \alpha'_{vk,w,sig} |sig\rangle \Bigg)$$

10. After applying $(U'_{\mathcal{O}_{\mathsf{OSS.Sign}},\mathsf{OSS.Sign}})^{-1}$ to $\mathsf{R_{vk}}, \mathsf{R_{msg}}, \mathsf{R_{sig}}, \mathsf{R_{aux,2}}$, we get

$$|1\rangle_{\mathsf{R_{msg}}} \otimes$$
$$\Bigg(\sum_{vk,w} \beta_{vk,w} \Bigg(|vk\rangle_{\mathsf{R_{vk}}} \otimes \Big(\sum_{z,o,w'} \gamma_{z,o,w'} |z\rangle_{\mathsf{R_{ora-inp,3}}} \Big| o \oplus \mathcal{O}^*_{\mathsf{OSS.GenQKey}(z)} \Big\rangle_{\mathsf{R_{ora-out,3}}} |H_0(vk,z)\rangle_{\mathsf{R_{ora-out,4}}}$$
$$\Big|0^\lambda\Big\rangle_{\mathsf{R_{ora-out,5}}} \otimes \big|w'\big\rangle_{\mathsf{R'_{aux,3}}} \Big) \Bigg) \otimes |\phi_{vk,w}\rangle_{\mathsf{R_{key}}} \otimes |w\rangle_{\mathsf{R_{out-aux,1}}} \otimes \Big|0^{p_2(\lambda)}\Big\rangle_{\mathsf{R_{aux,2}}} \Bigg)$$

11. After applying X to $\mathsf{R_{msg}}$, this register changes from $|1\rangle$ to $|0\rangle$, and the others are unaffected.

12. After applying $U'_{\mathcal{O}_{\mathsf{OSS.Sign}},\mathsf{OSS.Sign}}$ to $\mathsf{R_{crs}}, \mathsf{R_{vk}}, \mathsf{R_{key}}, \mathsf{R_{msg}}, \mathsf{R_{aux,2}}$, we get

$$|0\rangle_{\mathsf{R_{msg}}} \otimes$$
$$\Bigg(\sum_{vk,w} \beta_{vk,w} \Bigg(|vk\rangle_{\mathsf{R_{vk}}} \otimes \Big(\sum_{z,o,w'} \gamma_{z,o,w'} |z\rangle_{\mathsf{R_{ora-inp,3}}} \Big| o \oplus \mathcal{O}^*_{\mathsf{OSS.GenQKey}(z)} \Big\rangle_{\mathsf{R_{ora-out,3}}} |H_0(vk,z)\rangle_{\mathsf{R_{ora-out,4}}}$$
$$\Big|0^\lambda\Big\rangle_{\mathsf{R_{ora-out,5}}} \otimes \big|w'\big\rangle_{\mathsf{R'_{aux,3}}} \Big) \Bigg) \otimes \sum_{sig} \alpha_{vk,w,sig} |sig\rangle \Bigg)$$

where for each vk, w, sig with $\beta_{vk,w} \neq 0$ and $\alpha_{vk,w,sig} \neq 0$ we have that $\mathcal{O}^*_{\mathsf{OSS.Ver}}(crs^*, vk, sig, 0) = 1$ by correctness of $\mathsf{OSS.Sign}$.

13. After applying $U_{\mathcal{O}_0}$ with input registers $\mathsf{R_{vk}}, \mathsf{R_{sig}}, \mathsf{R_{ora-inp,3}}$ and the output register $\mathsf{R_{ora-out,4}}$, we get

$$|0\rangle_{\mathsf{R_{msg}}} \otimes$$
$$\Bigg(\sum_{vk,w} \beta_{vk,w} \Bigg(|vk\rangle_{\mathsf{R_{vk}}} \otimes \Big(\sum_{z,o,w'} \gamma_{z,o,w'} |z\rangle_{\mathsf{R_{ora-inp,3}}} \Big| o \oplus \mathcal{O}^*_{\mathsf{OSS.GenQKey}(z)} \Big\rangle_{\mathsf{R_{ora-out,3}}} \Big|0^\lambda\Big\rangle_{\mathsf{R_{ora-out,4}}}$$
$$\Big|0^\lambda\Big\rangle_{\mathsf{R_{ora-out,5}}} \otimes \big|w'\big\rangle_{\mathsf{R'_{aux,3}}} \Big) \Bigg) \otimes \sum_{sig} \alpha_{vk,w,sig} |sig\rangle \Bigg)$$

14. After applying $(U'_{\mathcal{O}_{\mathsf{OSS.Sign}},\mathsf{OSS.Sign}})^{-1}$ to $\mathsf{R_{vk}}, \mathsf{R_{msg}}, \mathsf{R_{sig}}, \mathsf{R_{aux,2}}$, we get

$$|0\rangle_{\mathsf{R_{msg}}} \otimes$$
$$\Bigg(\sum_{vk,w} \beta_{vk,w} \Bigg(|vk\rangle_{\mathsf{R_{vk}}} \otimes \Big(\sum_{z,o,w'} \gamma_{z,o,w'} |z\rangle_{\mathsf{R_{ora-inp,3}}} \Big| o \oplus \mathcal{O}^*_{\mathsf{OSS.GenQKey}(z)} \Big\rangle_{\mathsf{R_{ora-out,3}}} \Big|0^\lambda\Big\rangle_{\mathsf{R_{ora-out,4}}}$$
$$\Big|0^\lambda\Big\rangle_{\mathsf{R_{ora-out,5}}} \otimes \big|w'\big\rangle_{\mathsf{R'_{aux,3}}} \Big) \Bigg) \otimes |\phi_{vk,w}\rangle_{\mathsf{R_{key}}} \otimes |w\rangle_{\mathsf{R_{out-aux,1}}} \otimes \Big|0^{p_2(\lambda)}\Big\rangle_{\mathsf{R_{aux,2}}} \Bigg)$$

which is equal to

$$|\psi_{flame}\rangle \otimes |0\rangle_{\mathsf{R_{msg}}} \otimes \Big|0^{p_2(\lambda)}\Big\rangle_{\mathsf{R_{aux,2}}} \otimes U_2(U_1|\zeta\rangle) \otimes \Big|0^\lambda\Big\rangle_{\mathsf{R_{ora-out,4}}} \otimes \Big|0^\lambda\Big\rangle_{\mathsf{R_{ora-out,5}}} =$$
$$|\psi_{flame}\rangle \otimes |0\rangle_{\mathsf{R_{msg}}} \otimes \Big|0^{p_2(\lambda)}\Big\rangle_{\mathsf{R_{aux,2}}} \otimes U|\zeta\rangle \otimes \Big|0^\lambda\Big\rangle_{\mathsf{R_{ora-out,4}}} \otimes \Big|0^\lambda\Big\rangle_{\mathsf{R_{ora-out,5}}}$$

as claimed.

□

It is easy to see that the strong cloning correctness follows easily with this lemma: The loop starts with the state

$$|\psi\rangle_{\mathsf{R_{vk}},\mathsf{R_{key}},\mathsf{R_{out-aux,1}}} \otimes |0\rangle_{\mathsf{R_{msg}}} \otimes \left|0^{p_2(\lambda)}\right\rangle_{\mathsf{R_{aux,2}}} \otimes \left|0^{p_2(\lambda)}\right\rangle_{\mathsf{R_{aux,3}}} \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R_{ora-out,4}}} \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R_{ora-out,5}}},$$

thus, at the end, we get the state

$$\begin{aligned}
&|\psi_{flame}\rangle_{\mathsf{R_{vk}},\mathsf{R_{key}},\mathsf{R_{out-aux,1}}} \otimes |0\rangle_{\mathsf{R_{msg}}} \otimes \left|0^{p_2(\lambda)}\right\rangle_{\mathsf{R_{aux,2}}} \otimes (U^{p_1(\lambda)} \left|0^{p_2(\lambda)}\right\rangle)_{\mathsf{R_{aux,3}}} \otimes \\
&\left|0^{\lambda}\right\rangle_{\mathsf{R_{ora-out,4}}} \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R_{ora-out,5}}} = \\
&|\psi_{flame}\rangle_{\mathsf{R_{vk}},\mathsf{R_{key}},\mathsf{R_{out-aux,1}}} \otimes |0\rangle_{\mathsf{R_{msg}}} \otimes \left|0^{p_2(\lambda)}\right\rangle_{\mathsf{R_{aux,2}}} \otimes (|\psi\rangle)_{\mathsf{R_{aux,3}}} \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R_{ora-out,4}}} \otimes \left|0^{\lambda}\right\rangle_{\mathsf{R_{ora-out,5}}}.
\end{aligned}$$

Hence, the algorithm outputs $|\psi_{flame}\rangle \otimes |\psi_{flame}\rangle$.

10 Proof of Security of the Main Construction

In this section, we prove the security of our scheme. Throughout the proof, we will refer to the adversary that gets the flame state ($\mathcal{A}_1$) as the sender adversary and to the adversary that outputs the forged signature as the receiver adversary ($\mathcal{A}_2$).

Suppose for a contradiction that there exists a query-bounded adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that $\Pr\left[\mathsf{LR}-\mathsf{SIGN}-\mathsf{GAME}_{\mathcal{A}}(1^\lambda) = 1\right] > \frac{1}{q^*(\lambda)}$ for some polynomial $q^*(\cdot)$ and infinitely many values $\lambda > 0$. Note that $\mathcal{A}_1, \mathcal{A}_2$ will have query access to the oracles $\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}, \mathcal{O}_3, \mathcal{O}_4$ of the construction.

Adversary Assumptions Without loss of generality, we make the following assumptions about the adversaries $\mathcal{A}_1, \mathcal{A}_2$ (where $u(\lambda), t(\lambda)$ are some polynomials), and we also define some notation.

- $(\mathcal{A}_1, \mathcal{A}_2)$ runs for exactly $u(\lambda)$ rounds with probability 1, where each round consists of a message from $\mathcal{A}_1$ to $\mathcal{A}_2$ and then a message from $\mathcal{A}_2$ to $\mathcal{A}_1$ (except for the last round where $\mathcal{A}_2$ outputs $\perp$ as its message).
- We will denote the i-round of $\mathcal{A}_1$ as $\mathcal{A}_{1,i}$, for $i \in [u(\lambda)]$. That is, at round i, $\mathcal{A}_1$ simply executes $\mathcal{A}_{1,i}$ on its internal state register to obtain its message and the updated internal state register. $\mathcal{A}_{1,i}$ makes a total of $t(\lambda)$ queries to the oracles.
- Similar to above, we will denote the i-th round of $\mathcal{A}_2$ as $\mathcal{A}_{2,i}$ for $i \in [u(\lambda)]$. Note that for $\mathcal{A}_{2,u(\lambda)}$, the message output by it is $\perp$ since there is no message from $\mathcal{A}_2$ to $\mathcal{A}_1$. Finally, we will write $\mathcal{A}_{2,\mathsf{final}}$ to denote the challenge phase of the adversary $\mathcal{A}_2$: It receives the internal state register produced by $\mathcal{A}_{2,u(\lambda)}$ and the challenge message to be signed from the challenger and outputs a signature.
- Every round (i.e. for all $\ell \in [u(\lambda)]$), queries of $\mathcal{A}_{2,i}$ to the oracles $\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2$ proceeds in triples: It first makes one query to $\mathcal{O}_0$, then to $\mathcal{O}_1$, and then to $\mathcal{O}_2$. We will call these groups of three queries a *query-triplet*. Further, we assume that each round, $\mathcal{A}_{2,i}$ has exactly $t(\lambda)$ query-triplets, with any (polynomial) number of queries to the other oracles in between the triplets but with a total of exactly $t(\lambda)$ queries to each of the oracles $\mathcal{O}_3, \mathcal{O}_4, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_{\mathsf{OSS.Ver}}$ during the round. Note that this means that $\mathcal{A}_2$ has a total of $u(\lambda) \cdot t(\lambda)$ query-triplets and a total of $2u(\lambda) \cdot t(\lambda)$ over the leakage protocol.

Note that without loss of generality, we can take $t(\lambda) = 1$ by modifying our adversaries, since we do not impose an a-priori bound on the number of rounds. Similarly, without loss of generality, we can assume that each round, queries of $\mathcal{A}_{2,i}$ to $\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_3, \mathcal{O}_4$ occur in this order, after its queries to $\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2$.

Organization of the proof Our proof will be done through a sequence of hybrids, Hyb_0 (which is the original game $\mathsf{LR}-\mathsf{SIGN}-\mathsf{GAME}_{\mathcal{A}}(1^\lambda)$) through Hyb_3 (with some additional subhybrids in between). In the remainder of this section, we simply state our main theorems: We state our theorem about the indistinguishability of the first and the last hybrid (Theorem 19), and about the security in the final hybrid Hyb_3 (Theorem 20). Then, combining these main theorems simply proves the security of our scheme. The main theorems then are then proven in the subsequent subsections. In more detail, our proof is organized as follows.

In Section 10.1, we define various modified adversaries that we use in our hybrids. Then, in Section 10.2, we give the definitions of our hybrids. Then in Section 10.3, we prove Theorem 19,

that is, we show the indistinguishability of our hybrids (except for $\mathsf{Hyb}_{2,\ell,0}$,$\mathsf{Hyb}_{2,\ell,1}$ which we prove in Section 11). Finally, in Section 10.4, we prove Theorem 20, that is, we show the security in the final hybrid.

We now give our main theorem statements about the hybrids (which are defined in Section 10.2).

Theorem 19. $|\mathsf{Hyb}_0 - \mathsf{Hyb}_3| \leq \frac{1}{2 \cdot q^*(\lambda)}$

We prove this theorem in Section 10.3.

Theorem 20. $\Pr[\mathsf{Hyb}_3 = 1] \leq \mathsf{negl}(\lambda)$.

We prove this theorem in Section 10.4.

Now we complete the proof of security assuming these two theorems. We had assumed for a contradiction that $\Pr\left[\mathsf{LR-SIGN-GAME}_{\mathcal{A}}(1^\lambda) = 1\right] = \Pr[\mathsf{Hyb}_0 = 1] > \frac{1}{q^*(\lambda)}$ for some polynomial $q^*(\cdot)$ and infinitely many values $\lambda > 0$. Combining this with Theorem 19, we get that $\Pr[\mathsf{Hyb}_3 = 1] > \frac{1}{2q^*(\lambda)}$ for infinitely many $\lambda > 0$. However, this is a contradiction by Theorem 20. Therefore, our contradiction hypothesis is wrong, and thus we get $\Pr\left[\mathsf{LR-SIGN-GAME}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$, completing the proof of security.

10.1 Defining Modified Adversaries

In this section, we define the modified adversaries we will be using in the hybrids. We will explicitly denote the section queries to the oracles H_0, H_1.

We define $\mathcal{A}_{1,u(\lambda)+1}$ and $\mathcal{A}_{2,u(\lambda)+1}$ as follows.

$\underline{\mathcal{A}^{\mathcal{O}}_{1,u(\lambda)+1}(\mathsf{R}_{\mathsf{internal}}, S)}$

1. Output $\perp$.

$\underline{\mathcal{A}^{\mathcal{O}}_{2,u(\lambda)+1}(\mathsf{R}_{\mathsf{internal}}, L)}$

1. Sample $m' \leftarrow \mathcal{M}$.
2. $sig \leftarrow \mathcal{A}^{\mathcal{O}}_{2,\mathsf{final}}(\mathsf{R}_{\mathsf{internal}}, m')$.
3. Output $\perp, (m', sig)$.

We define $\mathcal{B}_{1,0}, \mathcal{B}_{2,0}$.

$\underline{\mathcal{B}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}_{1,0}(\mathsf{R})}$

1. Output $\mathsf{R}, \perp$.

$\underline{\mathcal{B}_{2,0}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(T)}$

1. Output $\perp$.

For $\ell \in [u(\lambda)+2]$, we define the adversaries $\mathcal{B}_{1,\ell}$,$\mathcal{B}_{2,\ell}$.

$\underline{\mathcal{B}_{1,\ell}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(\mathsf{R})}$

1. $\mathsf{R}_{\mathsf{internal}}, (T,S) \leftarrow \mathcal{B}_{1,\ell-1}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(\mathsf{R})$.
2. $\mathsf{R}'_{\mathsf{internal}}, L \leftarrow \mathcal{A}_{1,\ell}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(\mathsf{R}_{\mathsf{internal}}, S)$.
3. $V \leftarrow \mathcal{Q}_{1,\ell}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(T,L)$.
4. Set X_4 to be the empty set.
5. For $j \in [512^2 \cdot (t(\lambda))^6 \cdot \lambda^8 \cdot (u(\lambda))^4 \cdot (q^*(\lambda))^4 \cdot (\ell)^2]$:
    1. Sample a random index *ind* between 1 and the number of oracle queries made by $\mathcal{Q}_{2,\ell}$.
    2. Simulate $\mathcal{Q}_{2,\ell}^{\mathcal{O}_4}(V)$ until right before its *ind*-th oracle query is executed and measure the query input register instead of executing the query. Let m, y be the measurement outcome.
    3. Check if $\mathcal{O}_4(m,y)$ is 1. If so, add (m,y) to X_4.
6. Simulate $\mathcal{Q}_{2,\ell}(V)$ with the following oracle $\mathcal{O}'$: On input m, y, output 1 if $(m,y) \in X_4$, otherwise output 0. Let $(\mathsf{R}_{\mathsf{internal}}, S)$ be the output.
7. Set $T_{new} = (T, (V, X_4, S))$.
8. Output $\mathsf{R}'_{\mathsf{internal}}, (T_{new}, S)$.

$\underline{\mathcal{B}_{2,\ell}(T)}$

1. Parse $(T_{prev}, (V, X_4, S)) = T$.
2. Until $S' = S$:
    1. Simulate $\mathcal{Q}_{2,\ell}(V)$ with the following oracle $\mathcal{O}'$: On input m, y, output 1 if $(m,y) \in X_4$, otherwise output 0. Let $(\mathsf{R}_{\mathsf{internal}}, S')$ be the output.
3. Output $\mathsf{R}_{\mathsf{internal}}, S$.

$\mathcal{C}$-Adversaries We define $\mathcal{C}_{1,\ell,0,2}$ to be the following algorithm: On input (T, L), output $(T, L, (X_{0,0}, X_{1,0}, X_{2,0}, D_{0,0}, D_{1,0}, D_{2,0}))$ where $X_{0,0}, X_{1,0}, X_{2,0}$ are empty sets and $D_{0,0}, D_{1,0}, D_{2,0}$ are empty databases. We define $\mathcal{C}_{2,\ell,0,2}(Y)$ to be the algorithm that runs $\mathcal{A}_{2,\ell}(\mathsf{R}, L)$ after parsing $(T, L, Y') = Y$ and executing $\mathsf{R} \leftarrow \mathcal{B}_{2,\ell}(T)$.

For $\ell \in [u(\lambda)+1]$, $j \in [t(\lambda)]$ we define $\mathcal{C}_{1,\ell,j,0}$,$\mathcal{C}_{2,\ell,j,0}$.

$\mathcal{C}_{1,\ell,j,0}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(T,L)$

1. $Y \leftarrow \mathcal{C}_{1,\ell,j-1,2}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(T,L)$.
2. Parse $(T,\ L,\ ((X_{0,j'})_{j'\in[j-1]},\ (X_{1,j'})_{j'\in[j-1]},\ (X_{2,j'})_{j'\in[j-1]},\ (D_{0,j'})_{j'\in[j-1]},\ (D_{1,j'})_{j'\in[j-1]},\ (D_{2,j'})_{j'\in[j-1]})) = Y$.
3. Initialize $X_{0,j}$ to be the empty set and $D_{0,j}$ to be an empty set.
4. For $i \in [4194304 \cdot \lambda^{10} \cdot t^2(\lambda) \cdot (\ell)^5 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^4]$:
   1. Run $\mathcal{C}_{2,\ell,j-1,2}$ on input Y until its first query to $\mathcal{O}_0$ but do not execute the query. Instead, measure the query input register and let $ivk, isig, z$ be the measurement result. During this procedure[a], $\mathcal{C}_{2,\ell,j-1,2}$ makes queries to $\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_3, \mathcal{O}_4$ which are executed by forwarding them to the corresponding oracles.
   2. If $\mathcal{O}_0(ivk, isig, z) \neq \perp$, add $(ivk, isig)$ to $X_{0,j}$ and set $D_{0,j}(ivk||z) = \mathcal{O}_0(ivk, isig, z)$ for all z (*section query*).
5. Output $(T,\ L,\ ((X_{0,j'})_{j'\in[j]},\ (X_{1,j'})_{j'\in[j-1]},\ (X_{2,j'})_{j'\in[j-1]},\ (D_{0,j'})_{j'\in[j]},\ (D_{1,j'})_{j'\in[j-1]},\ (D_{2,j'})_{j'\in[j-1]}))$.

[a] Note that $\mathcal{C}_{1,\ell,j-1,2}$ does not make any queries to $\mathcal{O}_1, \mathcal{O}_2$ before making its first query to $\mathcal{O}_0$.

$\mathcal{C}_{2,\ell,j,0}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(Y)$

1. Parse $(T, L, ((X_{0,j'})_{j'\in[j]}, (X_{1,j'})_{j'\in[j-1]}, (X_{2,j'})_{j'\in[j-1]}, (D_{0,j'})_{j'\in[j]}, (D_{1,j'})_{j'\in[j-1]}, (D_{2,j'})_{j'\in[j-1]})) = Y$.
2. Set $Y' = (T,\ L,((X_{0,j'})_{j'\in[j-1]},\ (X_{1,j'})_{j'\in[j-1]},(X_{2,j'})_{j'\in[j-1]},\ (D_{0,j'})_{j'\in[j-1]},(D_{1,j'})_{j'\in[j-1]},\ (D_{2,j'})_{j'\in[j-1]}))$.
3. Run $\mathcal{C}_{2,\ell,j-1,2}$ on input Y' with the oracles $(\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_3, \mathcal{O}_4)$, but only the first query to $\mathcal{O}_0$ is instead simulated as follows: On input $ivk, isig, z$, output $D_{0,j}(ivk||z)$ if $(ivk, isig) \in X_{0,j}$ and otherwise output $\perp$. Let $\mathsf{R}_{\mathsf{internal}}, S$ be the output.
4. Output $\mathsf{R}_{\mathsf{internal}}, S$.

For $\ell \in [u(\lambda)+1]$, $j \in [t(\lambda)]$ we define $\mathcal{C}_{1,\ell,j,1}$,$\mathcal{C}_{2,\ell,j,1}$.

$\mathcal{C}_{1,\ell,j,1}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(T,L)$

1. $Y \leftarrow \mathcal{C}_{1,\ell,j,0}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(T,L)$.
2. Parse $(T, L, ((X_{0,j'})_{j'\in[j]}, (X_{1,j'})_{j'\in[j-1]}, (X_{2,j'})_{j'\in[j-1]}, (D_{0,j'})_{j'\in[j]}, (D_{1,j'})_{j'\in[j-1]}, (D_{2,j'})_{j'\in[j-1]})) = Y$.
3. Initialize $X_{1,j}$ to be the empty set and $D_{1,j}$ to be an empty set.

4. For $i \in [4194304 \cdot \lambda^{10} \cdot t^2(\lambda) \cdot (\ell)^5 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^4]$:
    1. Run $\mathcal{C}_{2,\ell,j,0}$ on input Y until its first query to $\mathcal{O}_1$ but do not execute the query. Instead, measure the query input register and let $ivk, isig$ be the measurement result. During this procedure[a], $\mathcal{C}_{2,\ell,j,0}$ makes queries to $\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_3, \mathcal{O}_4$ which are executed by forwarding them to the corresponding oracles.
    2. If $\mathcal{O}_1(ivk, isig, z) \neq \bot$, add $(ivk, isig)$ to $X_{1,j}$ and set $D_{1,j}(ivk||z) = \mathcal{O}_1(ivk, isig, z)$ for all z (*section query*).
5. Output $(T, L, (X_{0,j'})_{j' \in [j]}, (X_{1,j'})_{j' \in [j]}, (X_{2,j'})_{j' \in [j-1]}, (D_{0,j'})_{j' \in [j]}, (D_{1,j'})_{j' \in [j]}, (D_{2,j'})_{j' \in [j-1]})$.

[a] Note that $\mathcal{C}_{1,\ell,j,0}$ does not make any queries to $\mathcal{O}_0, \mathcal{O}_2$ before making its first query to $\mathcal{O}_1$.

$\underline{\mathcal{C}_{2,\ell,j,1}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(Y)}$

1. Parse $(T, L, ((X_{0,j'})_{j' \in [j]}, (X_{1,j'})_{j' \in [j]}, (X_{2,j'})_{j' \in [j-1]}, (D_{0,j'})_{j' \in [j]}, (D_{1,j'})_{j' \in [j]}, (D_{2,j'})_{j' \in [j-1]})) = Y$.
2. $Y' = (T, L, ((X_{0,j'})_{j' \in [j]}, (X_{1,j'})_{j' \in [j-1]}, (X_{2,j'})_{j' \in [j-1]}, (D_{0,j'})_{j' \in [j]}, (D_{1,j'})_{j' \in [j-1]}, (D_{2,j'})_{j' \in [j-1]}))$.
3. Run $\mathcal{C}_{2,\ell,j,0}$ on input Y' with the oracles $(\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_3, \mathcal{O}_4)$, but only the first query to $\mathcal{O}_1$ is instead simulated as follows: On input $ivk, isig, z$, output $D_{1,j}(ivk||z)$ if $(ivk, isig) \in X_{1,j}$ and otherwise output $\bot$. Let $\mathsf{R}_{\mathsf{internal}}, S$ be the output.
4. Output $\mathsf{R}_{\mathsf{internal}}, S$.

For $\ell \in [u(\lambda) + 1]$, $j \in [t(\lambda)]$ we define $\mathcal{C}_{1,\ell,j,2}$,$\mathcal{C}_{2,\ell,j,2}$.

$\underline{\mathcal{C}_{1,\ell,j,2}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(T, L)}$

1. $Y \leftarrow \mathcal{C}_{1,\ell,j,1}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(T, L)$.
2. Parse $(T, L, ((X_{0,j'})_{j' \in [j]}, (X_{1,j'})_{j' \in [j]}, (X_{2,j'})_{j' \in [j-1]}, (D_{0,j'})_{j' \in [j]}, (D_{1,j'})_{j' \in [j]}, (D_{2,j'})_{j' \in [j-1]})) = Y$.
3. Initialize $X_{2,j}$ to be the empty set and $D_{2,j}$ to be an empty set.
4. For $i \in [512 \cdot \lambda \cdot (t(\lambda) \cdot \ell) \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2]$:
    1. Run $\mathcal{C}_{2,\ell,j,1}$ on input Y until its first query to $\mathcal{O}_2$ but do not execute the query. Instead, measure the query input register and let ivk, y_0, y_1, z be the measurement result. During this procedure[a], $\mathcal{C}_{2,\ell,j,1}$ makes queries to $\mathcal{O}_{\mathsf{OSS.Sign}}, \mathcal{O}_3, \mathcal{O}_4$ which are executed by forwarding them to the corresponding oracles.
    2. If $\mathcal{O}_2(ivk, y_0, y_1, z) \neq \bot$, add (ivk, y_0, y_1, z) to $X_{2,j}$ and set $D_{2,j}(z) = \mathcal{O}_2(ivk, y_0, y_1, z)$.
5. Output $(T, L, (X_{0,j'})_{j' \in [j]}, (X_{1,j'})_{j' \in [j]}, (X_{2,j'})_{j' \in [j]}, (D_{0,j'})_{j' \in [j]}, (D_{1,j'})_{j' \in [j]}, (D_{2,j'})_{j' \in [j]})$.

[a] Note that $\mathcal{C}_{1,\ell,j,1}$ does not make any queries to $\mathcal{O}_0, \mathcal{O}_1$ before making its first query to $\mathcal{O}_2$.

$\underline{\mathcal{C}_{2,\ell,j,2}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(Y)}$

1. Parse $(T,\ L,((X_{0,j'})_{j'\in[j]},(X_{1,j'})_{j'\in[j]},(X_{2,j'})_{j'\in[j]},(D_{0,j'})_{j'\in[j]},(D_{1,j'})_{j'\in[j]},(D_{2,j'})_{j'\in[j]})) = Y$.
2. $Y' = (T, L,((X_{0,j'})_{j'\in[j]},(X_{1,j'})_{j'\in[j]},(X_{2,j'})_{j'\in[j-1]},(D_{0,j'})_{j'\in[j]},(D_{1,j'})_{j'\in[j]},(D_{2,j'})_{j'\in[j-1]}))$.
3. Run $\mathcal{C}_{2,\ell,j,1}$ on input Y' with the oracles $(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)$, but only the first query to $\mathcal{O}_2$ is instead simulated as follows: On input ivk, y_0, y_1, z, output $D_{2,j}(z)$ if $(ivk, y_0, y_1, z) \in X_{2,j}$ and otherwise output $\perp$. Let $\mathsf{R}_{\mathsf{internal}}, S$ be the output.
4. Output $\mathsf{R}_{\mathsf{internal}}, S$.

We define $\mathcal{C}_{1,\ell}$,$\mathcal{C}_{2,\ell}$ to be $\mathcal{C}_{1,\ell,t(\lambda),2}$ and $\mathcal{C}_{2,\ell,t(\lambda),2}$, respectively. For ease of reference, we write the full description of $\mathcal{C}_{2,\ell}$ with the internal calls unwound.

$\underline{\mathcal{C}_{2,\ell}^{\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4}(Y)}$

1. Parse $(T, L, ((X_{0,j'})_{j'\in[j]}, (X_{1,j'})_{j'\in[t(\lambda)]}, (X_{2,j'})_{j'\in[j]}, (D_{0,j'})_{j'\in[t(\lambda)]}, (D_{1,j'})_{j'\in[t(\lambda)]}, (D_{2,j'})_{j'\in[t(\lambda)]})) = Y$.
2. Run $\mathcal{B}_{2,\ell-1}(T)$ to get $\mathsf{R}_{\mathsf{internal}}$.
3. Simulate $\mathcal{A}_{2,\ell}^{(\mathcal{O}'_0,\mathcal{O}'_1,\mathcal{O}'_2,\mathcal{O}^*_{\mathsf{OSS.Sign}},\mathcal{O}^*_{\mathsf{OSS.Ver}},\mathcal{O}^*_3,\mathcal{O}^*_4)}(\mathsf{R}_{\mathsf{internal}}, L)$ where
   - The i-th (for $i \in [t(\lambda)]$) query-triplet to the oracles $\mathcal{O}'_0, \mathcal{O}'_1, \mathcal{O}'_2$ is executed with the oracles $\mathcal{O}^i_0, \mathcal{O}^i_1, \mathcal{O}^i_2$ defined as follows.
     - $\mathcal{O}^i_0(ivk, isig, z)$: If $(ivk, isig) \notin X_{0,i}$, output $\perp$. Otherwise, output $(D_{0,i}(ivk||z))$.
     - $\mathcal{O}^i_1(ivk, isig, z)$: If $(ivk, isig) \notin X_{1,i}$, output $\perp$. Otherwise, output $(D_{1,i}(ivk||z))$.
     - $\mathcal{O}^i_2(ivk, y_0, y_1, z)$: If $(ivk, y_0, y_1, z) \notin X_{2,i}$, output $\perp$. Otherwise, output $D_{2,i}(z)$.

   Let $\mathsf{R}'_{\mathsf{internal}}, S$ be the output.
4. Output $\mathsf{R}'_{\mathsf{internal}}, S$.

$\mathcal{R}$-Adversaries For $\ell \in [u(\lambda)+1]$, we define the adversaries $\mathcal{R}_{1,\ell}$,$\mathcal{R}_{2,\ell}$.

$\underline{\mathcal{R}_{1,\ell}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(T, L)}$

1. Set X_{sign} to be the empty set.
2. $Y \leftarrow \mathcal{C}_{1,\ell}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(T, L)$.
3. For $j \in [\lambda^8 \cdot \ell^2 \cdot (t(\lambda))^4 \cdot (n(\lambda))^2 \cdot (t(\lambda) \cdot u(\lambda) \cdot q^*(\lambda))^2 \cdot (\lambda \cdot u(\lambda) \cdot q^*(\lambda))^2]$:
   1. Sample a random index *ind* between 1 and the number of oracle queries by $\mathcal{C}_{2,\ell}$ to $\mathcal{O}_{\mathsf{OSS.Sign}}$.

2. Simulate $\mathcal{C}_{2,\ell}^{\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4}(Y)$ until right before its ind-th oracle query to $\mathcal{O}_{\mathsf{OSS.Sign}}$ is executed and measure the query input register instead of executing the query. Let vk, x be the measurement outcome. Check if $\mathcal{O}_{\mathsf{OSS.Sign}}(vk, x) \neq 0$ and if $x \notin \mathsf{Span}(\{v : (vk, v) \in X_{sign}\})$ and if so add (vk, x) to X_{sign}.

4. Output (Y, X_{sign}).

$\underline{\mathcal{R}_{2,\ell}^{\mathcal{O}_3,\mathcal{O}_4}(G)}$

1. Parse $(Y, X_{sign}) = G$.
2. Define the oracle $\mathcal{O}'_{sign}$: On input vk, z, output 1 if $z \notin \mathsf{Span}(\{v : (vk, v) \in X_{sign}\}) \setminus \{0\}$. Otherwise, output 0.
3. Simulate $\mathcal{C}_{2,\ell}^{\mathcal{O}'_{sign},\mathcal{O}_3,\mathcal{O}_4}(Y)$ to obtain $\mathsf{R}_{\mathsf{internal}}, S$.
4. Output $\mathsf{R}_{\mathsf{internal}}, S$.

$\mathcal{Q}$-Adversaries For $\ell \in [u(\lambda) + 1]$, we define the adversaries $\mathcal{Q}_{1,\ell}, \mathcal{Q}_{2,\ell}$.

$\underline{\mathcal{Q}_{1,\ell}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(T, L)}$

1. Set X_3 to be the empty list.
2. Set D_3 to be empty database.
3. $G \leftarrow \mathcal{R}_{1,\ell}^{(\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_{\mathsf{OSS.Sign}},\mathcal{O}_3,\mathcal{O}_4)}(T, L)$.
4. For $i \in [512 \cdot \lambda \cdot (t^2(\lambda) \cdot \ell) \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2]$:
    1. Sample a random index ind between 1 and the number of oracle queries by $\mathcal{R}_{2,\ell}$ to $\mathcal{O}_3$.
    2. Simulate $\mathcal{R}_{2,\ell}^{\mathcal{O}_3,\mathcal{O}_4}(G)$ until right before its ind-th oracle query to $\mathcal{O}_3$ is executed and measure the query input register instead of executing the query. Let ivk, y_0, y_1, m be the measurement outcome. If $\mathcal{O}_3(ivk, y_0, y_1, m) \neq \perp$, add (ivk, y_0, y_1, m) to X_3 and set $D_3(m) = \mathcal{O}_3(ivk, y_0, y_1, m)$.
5. Output $(G, (X_3, D_3))$.

$\underline{\mathcal{Q}_{2,\ell}^{\mathcal{O}_4}(V)}$

1. Parse $(G, (X_3, D_3)) = V$.
2. Define the oracle $\mathcal{O}'_3$: On input ivk, y_0, y_1, m, output $D_3(m)$ if $ivk, y_0, y_1, m \in X_3$, otherwise output $\perp$.
3. Simulate $\mathcal{R}_{2,\ell}^{\mathcal{O}'_3,\mathcal{O}_4}(G)$ to obtain $\mathsf{R}_{\mathsf{internal}}, S$.

4. Output $\mathsf{R}_{\mathsf{internal}}, S$.

10.2 Defining the Hybrids

In this section, we define the hybrids for our proof, each of which is constructed by modifying the previous one.

$\underline{\mathsf{Hyb}_0}$**:** The original game $\mathsf{LR}-\mathsf{SIGN}-\mathsf{GAME}_{\mathcal{A}}(1^\lambda)$ (Definition 24). For ease of reference, we write it out in full, with the notation introduced above.

$\underline{\mathsf{Hyb}_0}$

1. The challenger samples $sn^*, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QKeyFireSign.Setup}(1^\lambda)$.
2. Parse $\mathcal{O}^*_{\mathsf{QKeyFireSign}} = (\mathcal{O}^*_0, \mathcal{O}^*_1, \mathcal{O}^*_2, \mathcal{O}^*_{\mathsf{OSS.Sign}}, \mathcal{O}^*_3, \mathcal{O}^*_4) = sn^*$.
3. Set $\mathsf{R}^0_{1,\mathsf{internal}} = \mathsf{R}_{\mathsf{flame}}$ and $S^0 = \bot$.
4. Set $\mathsf{R}^0_{2,\mathsf{internal}} = \bot$.
5. For $i \in [u(\lambda)]$:
   1. $\mathcal{A}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,i}$ receives $\mathsf{R}^{i-1}_{1,\mathsf{internal}}$ and S^{i-1}, and it outputs an internal state register $\mathsf{R}^i_{1,\mathsf{internal}}$ and a leakage message L^i.
   2. $\mathcal{A}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{2,i}$ receives $\mathsf{R}^{i-1}_{2,\mathsf{internal}}$ and L^i, and it outputs an internal state register $\mathsf{R}^i_{2,\mathsf{internal}}$ and a message S^i.
6. The challenger samples a random message $m^* \leftarrow \mathcal{M}$.
7. The adversary $\mathcal{A}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{2,\mathsf{final}}$ receives $\mathsf{R}^{u(\lambda)}_{2,\mathsf{internal}}, m^*$, and it outputs sig^*.
8. Output $\mathcal{O}^*_4(m^*, sig^*)$.

$\underline{\mathsf{Hyb}_1}$**:** In this hybrid, we simply convert the challenge phase of the experiment into an additional round by using a special choice of adversaries (that is $\mathcal{A}_{1,\ell+1}, \mathcal{A}_{2,\ell+1}$) for this new round.

$\underline{\mathsf{Hyb}_1}$

1. The challenger samples $sn^*, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QKeyFireSign.Setup}(1^\lambda)$.
2. Parse $\mathcal{O}^*_{\mathsf{QKeyFireSign}} = (\mathcal{O}^*_0, \mathcal{O}^*_1, \mathcal{O}^*_2, \mathcal{O}^*_{\mathsf{OSS.Sign}}, \mathcal{O}^*_3, \mathcal{O}^*_4) = sn^*$.
3. Set $\mathsf{R}^0_{1,\mathsf{internal}} = \mathsf{R}_{\mathsf{flame}}$ and $S^0 = \bot$.
4. Set $\mathsf{R}^0_{2,\mathsf{internal}} = \bot$.
5. For $i \in [u(\lambda)+1]$:

1. $\mathcal{A}_{1,i}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $\mathsf{R}^{i-1}_{1,\mathsf{internal}}$ and S^{i-1}, and it outputs an internal state register $\mathsf{R}^{i}_{1,\mathsf{internal}}$ and a leakage message L^i.
2. $\mathcal{A}_{2,i}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $\mathsf{R}^{i-1}_{2,\mathsf{internal}}$ and L^i, and it outputs an internal state register $\mathsf{R}^{i}_{2,\mathsf{internal}}$ and a message S^i.

6. Parse $(m^*, sig^*) = S^{u(\lambda)+1}$.
7. Output $\mathcal{O}^*_4(m^*, sig^*)$.

For the subhybrids $\mathsf{Hyb}_{2,\ell,c}$ below, each index ℓ corresponds to a round of the leakage protocol between $\mathcal{A}_1$ and $\mathcal{A}_2$.

$\mathsf{Hyb}_{2,\ell,0}$ **for** $\ell \in [u(\lambda)+2]$**:** In this hybrid, the first $\ell-1$ rounds of the leakage protocol of $\mathcal{A}_1, \mathcal{A}_2$ are essentially collapsed into (i.e. simulated by) a single message between the two *new* adversaries $\mathcal{B}_{1,\ell-1}, \mathcal{B}_{2,\ell-2}$ (where the new receiver adversary $\mathcal{B}_{2,\ell-2}$ does not make any oracle calls), and then rounds ℓ through $u(\lambda)+1$ are executed as before. The very first hybrid $\mathsf{Hyb}_{2,1,0}$ will be equivalent to Hyb_1. We also label some parts of the experiment for ease of reference later on.

$\mathsf{Hyb}_{2,\ell,0}$

1. The challenger samples $sn^*, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QKeyFireSign.Setup}(1^\lambda)$.
2. Parse $\mathcal{O}^*_{\mathsf{QKeyFireSign}} = (\mathcal{O}^*_0, \mathcal{O}^*_1, \mathcal{O}^*_2, \mathcal{O}^*_{\mathsf{OSS.Sign}}, \mathcal{O}^*_3, \mathcal{O}^*_4) = sn^*$.

Round-collapsed Simulation for First $\ell-1$ Rounds:

3. Run $\mathcal{B}_{1,\ell-1}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}(\mathsf{R}_{\mathsf{flame}})$ to get $\mathsf{R}^{\ell-1}_{1,\mathsf{internal}}$ and $(T^{\ell-1}, S^{\ell-1})$.
4. Run $\mathcal{B}_{2,\ell-1}(T^{\ell-1})$ to get $\mathsf{R}^{\ell-1}_{2,\mathsf{internal}}$.

Original Protocol Continuation:

5. For $i \in \{\ell, \ldots, u(\lambda)+1\}$:
   1. $\mathcal{A}_{1,i}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $\mathsf{R}^{i-1}_{1,\mathsf{internal}}$ and S^{i-1}, and it outputs an internal state register $\mathsf{R}^{i}_{1,\mathsf{internal}}$ and a leakage message L^i.
   2. $\mathcal{A}_{2,i}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $\mathsf{R}^{i-1}_{2,\mathsf{internal}}$ and L^i, and it outputs an internal state register $\mathsf{R}^{i}_{2,\mathsf{internal}}$ and a message S^i.
6. Parse $(m^*, sig^*) = S^{u(\lambda)+1}$.
7. Output $\mathcal{O}^*_4(m^*, sig^*)$.

$\mathsf{Hyb}_{2,\ell,1}$ **for** $\ell \in [u(\lambda)+1]$**:** In this hybrid, we remove the oracle calls to $\mathcal{O}^*_0, \mathcal{O}^*_1, \mathcal{O}^*_2$ from the receiver adversary during round ℓ, by instead simulating it with a new leakage and a new receiver adversary $\mathcal{C}_{2,\ell}$.

$\mathsf{Hyb}_{2,\ell,1}$

1. The challenger samples $sn^*, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QKeyFireSign.Setup}(1^\lambda)$.
2. Parse $\mathcal{O}^*_{\mathsf{QKeyFireSign}} = (\mathcal{O}^*_0, \mathcal{O}^*_1, \mathcal{O}^*_2, \mathcal{O}^*_{\mathsf{OSS.Sign}}, \mathcal{O}^*_3, \mathcal{O}^*_4) = sn^*$.

   Round-collapsed Simulation for First $\ell - 1$ Rounds:

3. Run $\mathcal{B}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,\ell-1}(\mathsf{R}_{\mathsf{flame}})$ to get $\mathsf{R}^{\ell-1}_{1,\mathsf{internal}}$ and $(T^{\ell-1}, S^{\ell-1})$.

   Round ℓ:

4. $\mathcal{A}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,\ell}$ receives $\mathsf{R}^{\ell-1}_{1,\mathsf{internal}}$ and $S^{\ell-1}$, and it outputs an internal state register $\mathsf{R}^{\ell}_{1,\mathsf{internal}}$ and a leakage message L^ℓ
5. $\mathcal{C}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,\ell}$ receives $T^{\ell-1}, L^\ell$ and produces a message Y^ℓ.
6. Run $\mathcal{C}^{(\mathcal{O}^*_{\mathsf{OSS.Sign}}, \mathcal{O}^*_3, \mathcal{O}^*_4)}_{2,\ell}(Y^\ell)$ to obtain $(\mathsf{R}^{\ell}_{2,\mathsf{internal}}, S^\ell)$

   Original Protocol Continuation:

7. For $i \in \{\ell + 1, \ldots, u(\lambda) + 1\}$:
   1. $\mathcal{A}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,i}$ receives $\mathsf{R}^{i-1}_{1,\mathsf{internal}}$ and S^{i-1}, and it outputs a leakage message L^i and an internal state register $\mathsf{R}^{i}_{1,\mathsf{internal}}$.
   2. $\mathcal{A}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{2,i}$ receives $\mathsf{R}^{i-1}_{2,\mathsf{internal}}$ and L^i, and it outputs a message S^i and an internal state register $\mathsf{R}^{i}_{2,\mathsf{internal}}$.
8. Parse $(m^*, sig^*) = S^{u(\lambda)+1}$.
9. Output $\mathcal{O}^*_4(m^*, sig^*)$.

$\mathsf{Hyb}_{2,\ell,2}$ **for** $\ell \in [u(\lambda) + 1]$**:** We further remove the calls to $\mathcal{O}^*_{\mathsf{OSS.Sign}}$ made by the receiver adversary during this round, instead simulating it with a new leakage and new receiver adversary $\mathcal{R}_{2,\ell}$.

$\mathsf{Hyb}_{2,\ell,2}$

1. The challenger samples $sn^*, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QKeyFireSign.Setup}(1^\lambda)$.
2. Parse $\mathcal{O}^*_{\mathsf{QKeyFireSign}} = (\mathcal{O}^*_0, \mathcal{O}^*_1, \mathcal{O}^*_2, \mathcal{O}^*_{\mathsf{OSS.Sign}}, \mathcal{O}^*_3, \mathcal{O}^*_4) = sn^*$.

   Round-collapsed Simulation for First $\ell - 1$ Rounds:

3. Run $\mathcal{B}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,\ell-1}(\mathsf{R}_{\mathsf{flame}})$ to get $\mathsf{R}^{\ell-1}_{1,\mathsf{internal}}$ and $(T^{\ell-1}, S^{\ell-1})$.

   Round ℓ:

4. $\mathcal{A}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,\ell}$ receives $\mathsf{R}^{\ell-1}_{1,\mathsf{internal}}$ and $S^{\ell-1}$, and it outputs an internal state register $\mathsf{R}^{\ell}_{1,\mathsf{internal}}$ and a leakage message L^ℓ

5. $\mathcal{R}_{1,\ell}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $T^{\ell-1}, L^{\ell}$ and produces a message G^{ℓ}.
6. Run $\mathcal{R}_{2,\ell}^{\mathcal{O}^*_3,\mathcal{O}^*_4}(G^{\ell})$ to obtain $(\mathsf{R}^{\ell}_{2,\mathsf{internal}}, S^{\ell})$

Original Protocol Continuation:

7. For $i \in \{\ell+1, \ldots, u(\lambda)+1\}$:
   1. $\mathcal{A}_{1,i}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $\mathsf{R}^{i-1}_{1,\mathsf{internal}}$ and S^{i-1}, and it outputs a leakage message L^i and an internal state register $\mathsf{R}^{i}_{1,\mathsf{internal}}$.
   2. $\mathcal{A}_{2,i}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $\mathsf{R}^{i-1}_{2,\mathsf{internal}}$ and L^i, and it outputs a message S^i and an internal state register $\mathsf{R}^{i}_{2,\mathsf{internal}}$.
8. Parse $(m^*, sig^*) = S^{u(\lambda)+1}$.
9. Output $\mathcal{O}^*_4(m^*, sig^*)$.

$\mathsf{Hyb}_{2,\ell,3}$ **for** $\ell \in [u(\lambda)+1]$**:** We further remove the calls to $\mathcal{O}^*_3$ made by the receiver adversary during this round, instead simulating it with a new leakage and new receiver adversary $\mathcal{Q}_{2,\ell}$.

$\mathsf{Hyb}_{2,\ell,3}$

1. The challenger samples $sn^*, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QKeyFireSign.Setup}(1^{\lambda})$.
2. Parse $\mathcal{O}^*_{\mathsf{QKeyFireSign}} = (\mathcal{O}^*_0, \mathcal{O}^*_1, \mathcal{O}^*_2, \mathcal{O}^*_{\mathsf{OSS.Sign}}, \mathcal{O}^*_3, \mathcal{O}^*_4) = sn^*$.

Round-collapsed Simulation for First $\ell-1$ Rounds:

3. Run $\mathcal{B}_{1,\ell-1}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}(\mathsf{R}_{\mathsf{flame}})$ to get $\mathsf{R}^{\ell-1}_{1,\mathsf{internal}}$ and $(T^{\ell-1}, S^{\ell-1})$.

Round ℓ:

4. $\mathcal{A}_{1,\ell}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $\mathsf{R}^{\ell-1}_{1,\mathsf{internal}}$ and $S^{\ell-1}$, and it outputs an internal state register $\mathsf{R}^{\ell}_{1,\mathsf{internal}}$ and a leakage message L^{ℓ}
5. $\mathcal{Q}_{1,\ell}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $T^{\ell-1}, L^{\ell}$ and produces a message V^{ℓ}.
6. Run $\mathcal{Q}_{2,\ell}^{\mathcal{O}^*_4}(V^{\ell})$ to obtain $(\mathsf{R}^{\ell}_{2,\mathsf{internal}}, S^{\ell})$

Original Protocol Continuation:

7. For $i \in \{\ell+1, \ldots, u(\lambda)+1\}$:
   1. $\mathcal{A}_{1,i}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $\mathsf{R}^{i-1}_{1,\mathsf{internal}}$ and S^{i-1}, and it outputs a leakage message L^i and an internal state register $\mathsf{R}^{i}_{1,\mathsf{internal}}$.
   2. $\mathcal{A}_{2,i}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $\mathsf{R}^{i-1}_{2,\mathsf{internal}}$ and L^i, and it outputs a message S^i and an internal state register $\mathsf{R}^{i}_{2,\mathsf{internal}}$.

8. Parse $(m^*, sig^*) = S^{u(\lambda)+1}$.
9. Output $\mathcal{O}_4^*(m^*, sig^*)$.

$\mathsf{Hyb}_{2,\ell,4}$ **for** $\ell \in [u(\lambda)+1]$**:** We further remove the calls to $\mathcal{O}_4^*$ made by the receiver adversary during this round, and instead we simulate it with a new leakage.

$\mathsf{Hyb}_{2,\ell,4}$

1. The challenger samples $sn^*, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QKeyFireSign.Setup}(1^\lambda)$.
2. Parse $\mathcal{O}^*_{\mathsf{QKeyFireSign}} = (\mathcal{O}_0^*, \mathcal{O}_1^*, \mathcal{O}_2^*, \mathcal{O}^*_{\mathsf{OSS.Sign}}, \mathcal{O}_3^*, \mathcal{O}_4^*) = sn^*$.

   Round-collapsed Simulation for First $\ell-1$ Rounds:

3. Run $\mathcal{B}_{1,\ell-1}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}(\mathsf{R}_{\mathsf{flame}})$ to get $\mathsf{R}^{\ell-1}_{1,\mathsf{internal}}$ and $(T^{\ell-1}, S^{\ell-1})$.

   Round ℓ:

4. $\mathcal{A}_{1,\ell}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $\mathsf{R}^{\ell-1}_{1,\mathsf{internal}}$ and $S^{\ell-1}$, and it outputs an internal state register $\mathsf{R}^{\ell}_{1,\mathsf{internal}}$ and a leakage message L^ℓ
5. $\mathcal{Q}_{1,\ell}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $T^{\ell-1}, L^\ell$ and produces a message V^ℓ.
6. Set X_4^ℓ to be the empty set.
7. For $j \in [512^2 \cdot (t(\lambda))^6 \cdot \lambda^8 \cdot (u(\lambda))^4 \cdot (q^*(\lambda))^4 \cdot (\ell)^2]$:
   1. Sample a random index ind between 1 and the number of oracle queries made by $\mathcal{Q}_{2,\ell}$.
   2. Simulate $\mathcal{Q}_{2,\ell}^{\mathcal{O}_4^*}(V^\ell)$ until right before its ind-th oracle query is executed and measure the query input register instead of executing the query. Let m, y be the measurement outcome.
   3. Check if $\mathcal{O}_4^*(m, y)$ is 1. If so, add (m, y) to X_4^ℓ.
8. Simulate $\mathcal{Q}_{2,\ell}(V^\ell)$ with the following oracle $\mathcal{O}'$: On input m, y, output 1 if $(m, y) \in X$, otherwise output 0. Let $(\mathsf{R}^\ell_{2,\mathsf{internal}}, S^\ell)$ be the output.

   Original Protocol Continuation:

9. For $i \in \{\ell+1, \ldots, u(\lambda)+1\}$:
   1. $\mathcal{A}_{1,i}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $\mathsf{R}^{i-1}_{1,\mathsf{internal}}$ and S^{i-1}, and it outputs a leakage message L^i and an internal state register $\mathsf{R}^{i}_{1,\mathsf{internal}}$.
   2. $\mathcal{A}_{2,i}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $\mathsf{R}^{i-1}_{2,\mathsf{internal}}$ and L^i, and it outputs a message S^i and an internal state register $\mathsf{R}^{i}_{2,\mathsf{internal}}$.

10. Parse $(m^*, sig^*) = S^{u(\lambda)+1}$.
11. Output $\mathcal{O}_4^*(m^*, sig^*)$.

Hyb_3**:** We define this hybrid to be the same as $\mathsf{Hyb}_{2,u(\lambda)+2,0}$.

Hyb_3

1. The challenger samples $sn^*, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QKeyFireSign.Setup}(1^\lambda)$.
2. Parse $\mathcal{O}^*_{\mathsf{QKeyFireSign}} = (\mathcal{O}_0^*, \mathcal{O}_1^*, \mathcal{O}_2^*, \mathcal{O}^*_{\mathsf{OSS.Sign}}, \mathcal{O}_3^*, \mathcal{O}_4^*) = sn^*$.
3. Run $\mathcal{B}_{1,u(\lambda)+1}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}(\mathsf{R}_{\mathsf{flame}})$ to get $\mathsf{R}^{\ell-1}_{1,\mathsf{internal}}$ and $(T^{u(\lambda)+1}, S^{u(\lambda)+1})$.
4. Run $\mathcal{B}_{2,u(\lambda)+1}(T^{u(\lambda)+1})$ to get $\mathsf{R}^{u(\lambda)+1}_{2,\mathsf{internal}}$.
5. Parse $(m^*, sig^*) = S^{u(\lambda)+1}$.
6. Output $\mathcal{O}_4^*(m^*, sig^*)$.

10.3 Hybrid Indistinguishability (Proof of Theorem 19)

In this section, we prove the indistinguishability of the first and the last hybrid. That is, we prove Theorem 19.

First we give the hybrid indistinguishability lemmata.

Lemma 5. $\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1$.

Proof. This is straightforward by the definition of $\mathcal{A}_{1,u(\lambda)+1}, \mathcal{A}_{2,u(\lambda)+1}$. □

Lemma 6. $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_{2,1,0}$.

Proof. This is straightforward by the definition of $\mathcal{B}_{1,0}, \mathcal{B}_{2,0}$. □

Lemma 7. *For all* $\ell \in [u(\lambda)+1]$, $\left|\mathsf{Hyb}_{2,\ell,0} - \mathsf{Hyb}_{2,\ell,1}\right| \leq \frac{1}{\lambda \cdot u(\lambda) \cdot q^*(\lambda)}$.

We prove this lemma in Section 11.

Lemma 8. *For all* $\ell \in [u(\lambda)+1]$, $\left|\mathsf{Hyb}_{2,\ell,1} - \mathsf{Hyb}_{2,\ell,2}\right| \leq \frac{1}{\lambda \cdot u(\lambda) \cdot q^*(\lambda)}$.

We prove this lemma in Section 10.3.1.

Lemma 9. *For all* $\ell \in [u(\lambda)+1]$, $\left|\mathsf{Hyb}_{2,\ell,2} - \mathsf{Hyb}_{2,\ell,3}\right| \leq \frac{1}{\lambda \cdot u(\lambda) \cdot q^*(\lambda)}$.

We prove this lemma in Section 10.3.2.

Lemma 10. *For all* $\ell \in [u(\lambda)+1]$, $\left|\mathsf{Hyb}_{2,\ell,3} - \mathsf{Hyb}_{2,\ell,4}\right| \leq \frac{1}{\lambda \cdot u(\lambda) \cdot q^*(\lambda)}$.

We prove this lemma in Section 10.3.3.

Lemma 11. *For all* $\ell \in [u(\lambda)+1]$, $\mathsf{Hyb}_{2,\ell,4} \equiv \mathsf{Hyb}_{2,\ell+1,0}$.

Proof. Observe that, for the internal state, $\mathcal{B}_{2,\ell+1}$ produces the exact same mixed state as the one produced by $\mathcal{Q}_{2,\ell}$ in $\mathsf{Hyb}_{2,\ell,4}$ since it is conditioning on the same output message. □

Lemma 12. $\mathsf{Hyb}_{2,u(\lambda)+2,0} \equiv \mathsf{Hyb}_3$.

Proof. By definition. □

Finally, combining these lemmas gives us

$$|\mathsf{Hyb}_0 - \mathsf{Hyb}_3| \leq \frac{1}{4 \cdot q^*(\lambda)}$$

which proves Theorem 19.

10.3.1 Proof of Lemma 8

In this section, we prove Lemma 8. That is, we prove that for all $\ell \in [u(\lambda)+1]$, $\left|\mathsf{Hyb}_{2,\ell,1} - \mathsf{Hyb}_{2,\ell,2}\right| \leq \frac{1}{\lambda \cdot u(\lambda) \cdot q^*(\lambda)}$. Throughout the rest of the proof, fix any $\ell \in [u(\lambda)+1]$.

Note that the difference between the hybrids $\mathsf{Hyb}_{2,\ell,1}, \mathsf{Hyb}_{2,\ell,2}$ is that in $\mathsf{Hyb}_{2,\ell,1}$ we run $\mathcal{C}_{2,\ell}$ with the actual oracle $\mathcal{O}^*_{\mathsf{OSS.Sign}}$ whereas in $\mathsf{Hyb}_{2,\ell,2}$, we use $\mathcal{R}_{2,\ell}$ which simulates $\mathcal{C}_{2,\ell}$ with an estimated (by $\mathcal{R}_{\ell,1}$) oracle. We will use the incompressibility property of OSS (Definition 21) to show that this only makes a difference of $\frac{1}{\lambda \cdot u(\lambda) \cdot q^*(\lambda)}$ statistical difference. In this proof whenever we refer to incompressibility we will be referring to $\mathcal{O}_{\mathsf{OSS.Sign}}$ incompressibility of OSS. We make two notes about this hybrid. First, note that this hybrid is only needed for the LOCC leakage-resilience security: it can be removed in case of non-interactive security. Second, this hybrid indistinguishability can be proven directly using strong $\mathcal{O}_{\mathsf{OSS.Sign}}$ incompressibility of any OSS (the proof would be essentially the same as Section 11.1), but here we use regular incompressibility and the properties of our actual OSS scheme, rather than strong incompressibility.

Let $\gamma_{vk,z}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^\ell}$ be the total query weight of $\mathcal{C}_{2,\ell}(Y^\ell)$ to $\mathcal{O}^*_{\mathsf{OSS.Sign}}$ in $\mathsf{Hyb}_{2,\ell,1}$ on the input vk, z. We also define the following.

$$\alpha_{vk,z}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^\ell} = \begin{cases} \gamma_{vk,z}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^\ell}, \text{ if } \mathcal{O}^*_{\mathsf{OSS.Sign}}(vk,z) \neq 0 \\ 0, \text{ if } \mathcal{O}^*_{\mathsf{OSS.Sign}}(vk,z) = 0 \end{cases}$$

We also define $W^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^\ell,X_{sign}}$. Here, let X_{sign} denote the values sampled during the execution of $\mathcal{R}_{1,\ell}$, and let L_{sign} denote its projection onto its elements' first components (i.e. $L_{sign} = \{vk : (vk,z) \in X_{sign}\}$). Similarly, for any vk, let $S(vk)$ denote $\mathsf{Span}(\{v : (vk,v) \in X_{sign}\}) \setminus \{0\}$.

$$W^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^\ell,X_{sign}} := \sum_{\substack{vk,z: \\ z \notin S(vk)}} \alpha_{vk,z}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^\ell}$$

We now claim the following.

Claim 1. *For all sufficiently large $\lambda > 0$,*

$$\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^\ell,X_{sign}}[W^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^\ell,X_{sign}}] \leq \frac{1}{128 \cdot t(\lambda) \cdot (\lambda \cdot u(\lambda) \cdot q^*(\lambda))^2}.$$

Proof. We will prove this using the incompressibility property of OSS: Consider the following adversaries $\mathcal{P}_1$ and $\mathcal{P}_2$ for it. $\mathcal{P}_1$ simulates the hybrid $\mathsf{Hyb}_{2,\ell,1}$ until $\mathcal{C}_{1,\ell}$ produces its output Y^ℓ, and then outputs Y^ℓ. Then, $\mathcal{P}_2$ receives this value and runs $\mathcal{C}_{2,\ell}$ with it. When considering $\mathcal{P}_1, \mathcal{P}_2$ for incompressibility of OSS, we assume that $\mathcal{P}_1, \mathcal{P}_2$ are only oracle-adversaries for OSS, and the other oracles are just simulated, e.g. they sample H_0 in advance and both use this sample to simulate

the oracle outputs of $\mathcal{O}_0^*$. Observe that both $\mathcal{P}_1$ and $\mathcal{P}_2$ can perform their simulations with their respective oracles (e.g. $\mathcal{P}_2$ only queries $\mathcal{O}^*_{\mathsf{Sign}}, \mathcal{O}^*_{\mathsf{Ver}}$ and does not query $\mathcal{O}^*_{\mathsf{OSS.GenQKey}}$).

Let $\mathfrak{L}$ be the incompressibility list for the signing oracle as in Definition 21, and we know $|\mathfrak{L}| = s_{OSS,\ell}(\lambda) = 32768 \cdot \lambda^5 \cdot (\ell \cdot t(\lambda))^2 \cdot (t(\lambda) \cdot u(\lambda) \cdot q^*(\lambda))^2$ since $\mathcal{P}_1$ makes at most $s_{OSS,\ell}(\lambda)/2$ queries to $\mathcal{O}^*_{\mathsf{OSS.GenQKey}}$. The list depends on the oracle itself and the *leakage* Y^ℓ, but we will not write this dependence explicitly for conciseness.

$$W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^\ell, X_{sign}, \mathfrak{L}} := \sum_{\substack{vk,z: \\ z \notin S(vk) \\ vk \notin \mathfrak{L}}} \alpha_{vk,z}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^\ell}$$

$$W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^\ell, X_{sign}, \mathfrak{L}} := \sum_{\substack{vk,z: \\ z \notin S(vk) \\ vk \in \mathfrak{L}}} \alpha_{vk,z}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^\ell}$$

Now we claim

$$\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^\ell, X_{sign}, \mathfrak{L}}[W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^\ell, X_{sign}, \mathfrak{L}}] \leq \mathsf{negl}(\lambda).$$

To see this, consider the value

$$A = \sum_{\substack{vk,z: \\ vk \notin \mathfrak{L}}} \alpha_{vk,z}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^\ell}$$

Observe that with probability 1, this is greater than or equal to $W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^\ell, X_{sign}, \mathfrak{L}}$. Then, observe that $\mathbb{E}[A] \leq \mathsf{negl}(\lambda)$ by the incompressibility property of OSS.

Now we claim that for all sufficiently large $\lambda > 0$

$$\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^\ell, X_{sign}, \mathfrak{L}}[W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^\ell, X_{sign}, \mathfrak{L}}] \leq \frac{1}{256 \cdot t(\lambda) \cdot (\lambda \cdot u(\lambda) \cdot q^*(\lambda))^2}.$$

To prove this, we first define for $i \in [s_{OSS,\ell}(\lambda)]$,

$$Q_i^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^\ell, X_{sign}, \mathfrak{L}} := \sum_{\substack{vk,z: \\ z \notin S(vk) \\ vk = \mathsf{Item}_i(\mathfrak{L})}} \alpha_{vk,z}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^\ell}$$

We claim $\mathbb{E}[Q_i] \leq \frac{1}{256 \cdot s_{OSS,\ell}(\lambda) \cdot t(\lambda) \cdot (\lambda \cdot u(\lambda) \cdot q^*(\lambda))^2}$ for all $i \in [s_{OSS,\ell}(\lambda)]$. Observe that with probability 1 we have $W = W_1 + \sum_{i \in [s_{OSS,\ell}(\lambda)]} Q_i$, thus, proving this claim will complete our proof.

Now fix some $i^* \in [s_{OSS,\ell}(\lambda)]$. Fix any value $\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^\ell$. We will prove the above for any such fixed values. Note that fixing these also fixes $\mathfrak{L}$ and thus $\mathsf{Item}_{i^*}(\mathfrak{L})$ - denote this as vk^*. There are $r_{OSS,\ell}(\lambda) = \lambda^8 \cdot \ell^2 \cdot (t(\lambda))^4 \cdot (n(\lambda))^2 \cdot (t(\lambda) \cdot u(\lambda) \cdot q^*(\lambda))^2 \cdot (\lambda \cdot u(\lambda) \cdot q^*(\lambda))^2$ iterations during the run of $\mathcal{R}_{\ell,1}$, and we will consider these iterations in $n(\lambda) + 1$ *groups*: we have $\frac{r_{OSS,\ell}(\lambda)}{n(\lambda)+1}$ iterations in each group. Now let GOOD denote the event that $Q_{i^*} \leq \frac{1}{256 \cdot \lambda \cdot s_{OSS,\ell}(\lambda) \cdot t(\lambda) \cdot (\lambda \cdot u(\lambda) \cdot q^*(\lambda))^2}$. We will prove that $\Pr[\mathsf{GOOD}] \geq 1 - 2\exp\{-\lambda\}$. For each $j \in [n+1]$, define the event E_j to be the event that during the j-th group, a new element is added to X_{sign}, and define E'_j to be the event that $\sum_{\substack{vk,z: \\ z \notin S_j(vk^*)}} \alpha_{vk^*,z}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^\ell} > \frac{1}{256 \cdot \lambda \cdot s_{OSS,\ell}(\lambda) \cdot t(\lambda) \cdot (\lambda \cdot u(\lambda) \cdot q^*(\lambda))^2}$ where S_j denotes $\mathsf{Span}(\{v : (vk, v) \in$

$X_{sign}\}) \setminus \{0\}$ for the value of X_{sign} at the beginning of the j-th group. It is easy to see that $\left(\vee_{j=1}^{n+1}\overline{E'_j}\right) \implies \mathsf{GOOD}$. Similarly, it is easy to see that $E_1 \wedge \cdots \wedge E_n \implies \overline{E'_{n+1}}$, since the subspace $\mathsf{Colspan}(A_{vk^*})^\perp$ is of dimension at most n, so, once n linearly independent elements are added, there cannot be any vectors outside $\mathsf{Colspan}(A_{vk^*})^\perp$. Note that combining these two, we also get $E_1 \wedge \cdots \wedge E_n \implies \mathsf{GOOD}$. Finally, we claim that $\Pr\left[E_j | E'_j \wedge E\right] \geq 1 - \exp\{-\lambda \cdot 2n\}$ for all $j \in [n]$ and any event E such that $\Pr[E] > 0$. For now, we will assume this and complete the proof, and later we will prove it. First, observe that $\Pr\left[\bigwedge_{j=1}^n E_j | \bigwedge_{j=1}^n E'_j\right] \geq (1-\exp\{-\lambda \cdot 2n\})^n \geq 1-\exp\{-\lambda\}$ by above. This also means $\Pr\left[\mathsf{GOOD} | \bigwedge_{j=1}^n E'_j\right] \geq 1-\exp\{-\lambda\}$. Also recall that $\overline{\bigwedge_{j=1}^n E'_j} \implies \mathsf{GOOD}$. Combining these two gives us $\Pr[\mathsf{GOOD}] \geq 1 - 2\exp\{-\lambda\}$.

Finally, we prove $\Pr\left[E_j | E'_j \wedge E\right] \geq 1 - \exp\{-\lambda \cdot 2n\}$ for all $j \in [n]$ and any event E such that $\Pr[E] > 0$. To see this, observe that during the j-th group, we measure the query of the adversary for $\frac{r_{OSS,\ell}(\lambda)}{n(\lambda)+1}$ times. Since the weight on *good elements* (i.e. elements that would be added to our list and thus cause E_j to occur) is at least $\frac{1}{256 \cdot \lambda \cdot s_{OSS,\ell}(\lambda) \cdot t(\lambda) \cdot (\lambda \cdot u(\lambda) \cdot q^*(\lambda))^2}$ when conditioned on E'_j, we get the desired result since $r_{OSS,\ell}(\lambda) \geq 1024 \cdot (n^2(\lambda) \cdot t(\lambda)) \cdot \lambda^2 \cdot s_{OSS,\ell}(\lambda) \cdot t(\lambda) \cdot (\lambda \cdot u(\lambda) \cdot q^*(\lambda))^2$. $\square$

Now we claim that the lemma follows from Claim 1. To see this, observe that the two hybrids only differ on the oracles they use for $\mathcal{C}_{\ell,2}$: $\mathsf{Hyb}_{2,\ell,1}$ uses $\mathcal{O}^*_{\mathsf{OSS.Sign}}$ whereas $\mathsf{Hyb}_{2,\ell,2}$ uses the estimated oracle created by $\mathcal{R}_{\ell,1}$. Observe that these oracles can differ only on inputs vk, z such that $\mathcal{O}^*_{\mathsf{OSS.Sign}}(vk, z) \neq 0$ (i.e. $z \in \mathsf{Colspan}(A_{vk})^\perp \setminus \{0\}$) but $z \notin S(vk)$. Now observe that $W^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^\ell, X_{sign}}$ is an upper bound on the total weight on the inputs on which the oracles might differ. Thus, the lemma follows by the Jensen's inequality, the fact that $\mathcal{C}_{2,\ell}$ makes at most $t(\lambda)$ queries to $\mathcal{O}^*_{\mathsf{OSS.Sign}}$ and Theorem 6.

10.3.2 Proof of Lemma 9

In this section, we prove Lemma 9. That is, we prove that for all $\ell \in [u(\lambda)+1]$, $\left|\mathsf{Hyb}_{2,\ell,2} - \mathsf{Hyb}_{2,\ell,3}\right| \leq \frac{1}{\lambda \cdot u(\lambda) \cdot q^*(\lambda)}$. Throughout the rest of the proof, fix any $\ell \in [u(\lambda)+1]$.

Observe that the difference between the two hybrids is that in $\mathsf{Hyb}_{2,\ell,2}$ we run $\mathcal{R}_{2,\ell}$ directly, whereas in $\mathsf{Hyb}_{2,\ell,3}$ we instead run $\mathcal{Q}_{2,\ell}$ which simulates $\mathcal{R}_{2,\ell}$ while using an estimated oracle for its queries to $\mathcal{O}^*_3$. We will use the incompressibility property of H_0, H_1 to show that this only makes a difference of $\frac{1}{\lambda \cdot u(\lambda) \cdot q^*(\lambda)}$ in statistical distance.

Let $\gamma^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, G^\ell}_{ivk, y_0, y_1, z}$ denote the total query weight of $\mathcal{R}_{2,\ell}(G^\ell)$ to the oracle $\mathcal{O}^*_2$ on input ivk, y_0, y_1, z. We also define the following.

$$\alpha^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, G^\ell}_{ivk,z} = \sum_{y_0, y_1 : \mathcal{O}^*_2(ivk, y_0, y_1, z) \neq \perp} \gamma^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, G^\ell}_{ivk, y_0, y_1, z}$$

Now let $(X^{\ell'}_{0,j'})_{\ell' \in [\ell], j' \in [t(\lambda)]}$, $(X^{\ell'}_{1,j'})_{\ell' \in [\ell], j' \in [t(\lambda)]}$, $(D^{\ell'}_{0,j'})_{\ell' \in [\ell^*], j' \in [i^*]}$, $(D^{\ell'}_{1,j'})_{\ell' \in [\ell^*], j' \in [i^*]}$ be the estimated lists and databases in V^ℓ (or equivalently, G^ℓ) when it is fully unwrapped and parsed. Let X_3, D_3 be the estimated list and database inside V^ℓ that is sampled by $\mathcal{Q}_{1,\ell}$. With abuse of notation, sometimes we will refer to the projection of the sets $X^{\ell'}_{\cdot,j'}$ onto their elements' (which are tuples) subcomponents also as $X^{\ell'}_{\cdot,j'}$ (e.g. we will write $X^{\ell'}_{2,j'}$ to mean the set of pairs (ivk, z) obtained by projecting each tuple in the actual set $X^{\ell'}_{2,j'}$ to its first and fourth elements) - however, the meaning will clear from context. Now we claim the following.

Claim 2.

$$\Pr\left[\left(\bigcup_{\ell'\in[\ell],j'\in[t(\lambda)]} X^{\ell'}_{0,j'}\right)\bigcap\left(\bigcup_{\ell'\in[\ell],j'\in[t(\lambda)]} X^{\ell'}_{1,j'}\right)=\emptyset\right]\geq 1-\mathsf{negl}(\lambda).$$

Proof. Observe that if this intersection is not empty, then there is ivk such that $ivk\in X^{\ell'}_{0,j'}$ and $ivk\in X^{\ell''}_{1,j''}$ for some ℓ',ℓ'',j',j''. However, by construction we know that this implies that we can obtain $isig_0, isig_1$ from V^ℓ such that $\mathcal{O}_{\mathsf{OSS.Ver}^*}(ivk,0,isig_0)=\mathcal{O}_{\mathsf{OSS.Ver}^*}(ivk,1,isig_1)=1$. Since V^ℓ is obtained by making polynomially many queries to the oracles of OSS, the claim follows by the one-shot security of OSS. □

Now we define the following (some dependencies are implicit to keep the notation concise).

$$W_0^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},G^\ell,X_3}=\sum_{\substack{ivk,z:\\ ivk\notin\left(\bigcup_{\ell'\in[\ell],j'\in[t(\lambda)]} X^{\ell'}_{0,j'}\right)\\ (ivk,z)\notin X_3}}\alpha_{ivk,z}$$

$$W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},G^\ell,X_3}=\sum_{\substack{ivk,z:\\ ivk\notin\left(\bigcup_{\ell'\in[\ell],j'\in[t(\lambda)]} X^{\ell'}_{1,j'}\right)\\ (ivk,z)\notin X_3}}\alpha_{ivk,z}$$

$$W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},G^\ell,X_3}=\sum_{\substack{ivk,z:\\ ivk\notin\left(\bigcup_{\ell'\in[\ell],j'\in[t(\lambda)]} X^{\ell'}_{0,j'}\right)\bigcap\left(\bigcup_{\ell'\in[\ell],j'\in[t(\lambda)]} X^{\ell'}_{1,j'}\right)}}\alpha_{ivk,z}$$

Now we make the following claims.

Claim 3.

$$\mathbb{E}[W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},G^\ell,X_3}]\leq\mathsf{negl}(\lambda). \tag{1}$$

Proof. This simply follows from the definition of W_2 and Claim 2. □

Claim 4. *For all sufficiently large $\lambda>0$.*

$$\mathbb{E}[W_0^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},G^\ell,X_3}]\leq\frac{1}{128\cdot\lambda^4\cdot(u(\lambda)\cdot q^*(\lambda)\cdot t(\lambda))^2}$$

$$\mathbb{E}[W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},G^\ell,X_3}]\leq\frac{1}{128\cdot\lambda^4\cdot(u(\lambda)\cdot q^*(\lambda)\cdot t(\lambda))^2}$$

Proof. We will only prove the first one (using incompressibility of the random oracle H_0), and the second bound follows by the same argument (but this time relying on the incompressibility of H_1).

Now we will use the incompressibility property (Theorem 15) of the random oracle H_0 with the predicate P defined as $P(ivk||z)=1$ if $ivk\in S$, where $S=\left(\bigcup_{\ell'\in[\ell],j'\in[t(\lambda)]} X^{\ell'}_{0,j'}\right)$. Note that this set S is exactly the set of prefixes on which our adversary has *section queries* (otherwise it only makes normal queries and verification queries). In more detail, let $\mathcal{P}_1$ be the adversary that simulates $\mathsf{Hyb}_{2,\ell,2}$ until the point where $\mathcal{R}_{1,\ell}$ outputs its message, and it also outputs S as its predicate. Then $\mathcal{P}_2$ simulates $\mathcal{R}_{2,\ell}$. We invoke Theorem 15 for this pair of adversaries, and let $\mathfrak{L}$

denote the incompressibility list that is promised to exist, and we know $|\mathfrak{L}| = \ell \cdot t(\lambda)$ since $\mathcal{P}_1$ makes at most $\ell \cdot t(\lambda)$ normal queries. We then split W_0 into two summands U_0, U_1:

$$U_0^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, G^\ell, X_3} = \sum_{\substack{ivk,z: \\ ivk \notin \left(\bigcup_{\ell' \in [\ell], j' \in [t(\lambda)]} X^{\ell'}_{0,j'}\right) \\ (ivk,z) \notin X_3 \\ (ivk,z) \notin \mathfrak{L}}} \alpha_{ivk,isig}$$

$$U_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, G^\ell, X_3} = \sum_{\substack{ivk,z: \\ ivk \notin \left(\bigcup_{\ell' \in [\ell], j' \in [t(\lambda)]} X^{\ell'}_{0,j'}\right) \\ (ivk,z) \notin X_3 \\ (ivk,z) \in \mathfrak{L}}} \alpha_{ivk,isig}$$

By the incompressibility lemma, we get $\mathbb{E}[U_0] \leq \mathsf{negl}(\lambda)$. Now, we will focus on bounding U_1.

First we claim that, for all $\mathcal{O}^*_{\mathsf{QKeyFireSign}}, G^\ell$, with overwhelming probability we have that any ivk, z that satisfies

$$\alpha_{ivk,z} \geq \frac{1}{512 \cdot (t(\lambda) \cdot \ell) \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$

is contained in X_3. To see this, observe that X_3 is created by measuring a random query of $\mathcal{R}_{\ell,2}$ to $\mathcal{O}_3$ for $512 \cdot \lambda \cdot (t^2(\lambda) \cdot \ell) \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2$ times. Thus, for any ivk, z such that $\alpha_{ivk,z}$ satisfies the above condition, the probability that $(ivk, z) \notin X_3$ is at most $\exp\{-\lambda\}$. Now define GOOD to be the event that all $(ivk, z) \in \mathfrak{L}$ that satisfies

$$\alpha_{ivk,z} \geq \frac{1}{512 \cdot (t(\lambda) \cdot \ell) \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$

are included in X_3. Since this is true with probability $1 - \exp\{-\lambda\}$ for each $ivk, z \in \mathfrak{L}$ individually, and since $|\mathfrak{L}|$ is polynomial, we have that $\Pr[\mathsf{GOOD}] \geq 1 - \mathsf{negl}(\lambda)$.

Now we claim

$$\mathbb{E}[U_1 \big| \mathsf{GOOD}] \leq \frac{1}{512 \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$

To see this, first by linearity of expectation we have

$$\mathbb{E}[U_1 \big| \mathsf{GOOD}] = \sum_{\substack{ivk,z: \\ ivk \notin \left(\bigcup_{\ell' \in [\ell], j' \in [t(\lambda)]} X^{\ell'}_{0,j'}\right) \\ (ivk,z) \notin X_3 \\ (ivk,z) \in \mathfrak{L}}} \mathbb{E}[\alpha_{ivk,z} | \mathsf{GOOD}]$$

However, by definition of GOOD, when GOOD occurs, we have

$$ivk, z \in \mathfrak{L} \setminus X_3 \implies \alpha_{ivk,z} \leq \frac{1}{512 \cdot (t(\lambda) \cdot \ell) \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$

Since $|\mathfrak{L}| = \ell \cdot t(\lambda)$, we obtain the desired upper bound on expectation conditioned on GOOD. Since $\Pr[\mathsf{GOOD}] \geq 1 - \mathsf{negl}(\lambda)$, we also get

$$\mathbb{E}[U_1] \leq \frac{1}{256 \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$

as desired. □

Observe that Lemma 16 follows from the above claims. To see this, observe that $W_0 + W_1 + W_2$ is an upper bound on the total weight on which the two different oracles in the hybrids might differ. Thus the result follows by the linearity of expectation, Theorem 6 and Jensen's inequality.

10.3.3 Proof of Lemma 10

In this section, we prove Lemma 10. That is, we prove that for all $\ell \in [u(\lambda)+1]$, $\left|\mathsf{Hyb}_{2,\ell,3} - \mathsf{Hyb}_{2,\ell,4}\right| \leq \frac{1}{\lambda \cdot u(\lambda) \cdot q^*(\lambda)}$. Throughout the rest of the proof, fix any $\ell \in [u(\lambda)+1]$.

First, note that the difference between the hybrids $\mathsf{Hyb}_{2,\ell,3}, \mathsf{Hyb}_{2,\ell,4}$ is that in $\mathsf{Hyb}_{2,\ell,3}$ we run $\mathcal{Q}_{2,\ell}$ with the actual oracle $\mathcal{O}_4^*$ whereas in $\mathsf{Hyb}_{2,\ell,4}$ we use the simulated oracle $\mathcal{O}'$. We will use the incompressibility property of the random oracle H_{sig} (Theorem 15) to show that this only makes a difference of $\frac{1}{\lambda \cdot u(\lambda) \cdot q^*(\lambda)}$ statistical difference. In this proof whenever we refer to incompressibility we will be referring to that of H_{sig}.

For the incompressibility property, consider the following adversaries $\mathcal{P}_1$ and $\mathcal{P}_2$. $\mathcal{P}_1$ simulates the hybrid $\mathsf{Hyb}_{2,\ell,3}$ until $\mathcal{Q}_{1,\ell}$ produces its output V^ℓ, and then outputs V^ℓ. Then, $\mathcal{P}_2$ receives this value and runs $\mathcal{Q}_{2,\ell}$ with it. When considering $\mathcal{P}_1, \mathcal{P}_2$ for incompressibility of H_{sig}, we assume that $\mathcal{P}_1, \mathcal{P}_2$ are only oracle-adversaries for H_{sig} and the verifier (as in Theorem 15) of H_{sig} respectively, and the other oracles are just simulated, e.g. they sample H_0 in advance and both use this sample to simulate the oracle calls to $\mathcal{O}_0^*$. Observe that both $\mathcal{P}_1$ and $\mathcal{P}_2$ can perform their simulations with their respective oracles, e.g., $\mathcal{O}_4^*$ is simply the verification oracle of H_{sig} and $\mathcal{P}_2$ does not query H_{sig}.

Let $\mathfrak{L}$ be the incompressibility list as in Theorem 15, and we know $|\mathfrak{L}| = s_{4,\ell}(\lambda) = 512 \cdot \lambda \cdot (t^2(\lambda) \cdot \ell^2(\lambda)) \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2$ since that many queries to H_{sig} are made by $\mathcal{P}_1$. The list depends on the oracle itself and the *leakage* V^ℓ, but we will not write this dependence explicitly for conciseness.

Let $\gamma_{m,y}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, V^\ell}$ be the total query weight of $\mathcal{Q}_{2,\ell}(V^\ell)$ to $\mathcal{O}_4^*$ in $\mathsf{Hyb}_{2,\ell,2}$ on the input m, y. Define α_m as

$$\alpha_m^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, V^\ell} = \sum_{y : \mathcal{O}_4^*(m,y)=1} \gamma_{m,y}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, V^\ell}.$$

Now define $W^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, V^\ell, X_4^\ell}, W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, V^\ell, X_4^\ell, \mathfrak{L}}, W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, V^\ell, X_4^\ell, \mathfrak{L}}$ as

$$W^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, V^\ell, X_4^\ell} := \sum_{\substack{m: \\ m \notin X_4^\ell}} \alpha_m^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, V^\ell}$$

$$W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, V^\ell, X_4^\ell, \mathfrak{L}} := \sum_{\substack{m: \\ m \notin X_4^\ell \\ m \notin \mathfrak{L}}} \alpha_m^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, V^\ell}$$

$$W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, V^\ell, X_4^\ell, \mathfrak{L}} := \sum_{\substack{m: \\ m \notin X_4^\ell \\ m \in \mathfrak{L}}} \alpha_m^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, V^\ell}$$

In the definition above, and in the rest of the proof, by abuse of notation, we write $m \in X_4^\ell$ to denote that there exists a pair $(m, y) \in X_4^\ell$ for some y.

We claim the following.

Claim 5. *For all sufficiently large $\lambda > 0$,*

$$\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, V^\ell, X_4^\ell, \mathfrak{L}}[W^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, V^\ell, X_4^\ell}] \leq \frac{1}{128 \cdot t(\lambda) \cdot (\lambda \cdot u(\lambda) \cdot q^*(\lambda))^2}.$$

Observe that once we prove this, the lemma follows by Theorem 6 and by the Jensen's inequality since the oracles $\mathcal{O}_4^*$ and $\mathcal{O}'$ in these hybrids can possibly differ only on inputs (m,y) such that $(m,y) \notin X_4^\ell$ and $\mathcal{O}_4^*(m,y)=1$, and $W^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell}$ is an upper bound on the total weight on such inputs and $\mathcal{Q}_{2,\ell}$ makes at most $t(\lambda)$ queries to the oracle $\mathcal{O}_4^*$.

To prove Claim 5, we make the following two other claims.

Claim 6.

$$\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}}[W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}}] \leq \mathsf{negl}(\lambda).$$

Proof. Consider the value.

$$A = \sum_{\substack{m:\\ m\notin\mathfrak{L}}} \alpha_m^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell}$$

Observe that this value is always greater than or equal to $W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}}$. Also observe that $\mathbb{E}[A] \leq \mathsf{negl}(\lambda)$ by the incompressibility of H_{sig}. □

Claim 7. *For all sufficiently large $\lambda > 0$:*

$$\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}}[W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}}] \leq \frac{1}{256\cdot t(\lambda)\cdot(\lambda\cdot u(\lambda)\cdot q^*(\lambda))^2}.$$

Proof. First we claim that, for all $\mathcal{O}^*_{\mathsf{QKeyFireSign}}, V^\ell$, with overwhelming probability we have that any $m\in\mathcal{M}$ that satisfies

$$\alpha_m^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell} \geq \frac{1}{512\cdot t(\lambda)\cdot(\lambda\cdot u(\lambda)\cdot q^*(\lambda))^2\cdot s_{4,\ell}(\lambda)}$$

is contained in X_4^ℓ. To see this, observe that X_4^ℓ is obtained by measuring a random query of $\mathcal{Q}_{2,\ell}(V^\ell)$ to $\mathcal{O}_4^*$ for $r_{4,\ell}(\lambda) = 512^2\cdot(t(\lambda))^6\cdot\lambda^8\cdot(u(\lambda))^4\cdot(q^*(\lambda))^4\cdot(\ell)^2$ times and adding to the set if the pair m,y obtained from the measurement is *valid* (i.e. $\mathcal{O}_4^*(m,y)=1$). Thus, for any $m\in\mathcal{M}$ that satisfies the above condition, the probability that $m\notin X_4^\ell$ is at most $\exp\left\{-r_{4,\ell}(\lambda)\cdot\frac{\alpha_m^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell}}{t(\lambda)}\right\}$. Since we have

$$1-\exp\left\{r_{4,\ell}(\lambda)\cdot\frac{1}{t(\lambda)\cdot 512\cdot t(\lambda)\cdot(\lambda\cdot u(\lambda)\cdot q^*(\lambda))^2\cdot s_{4,\ell}(\lambda)}\right\} = 1-\exp\{-\lambda\}$$

this claim follows.

Now define GOOD to be the event that all $m\in\mathfrak{L}$ that satisfies

$$\alpha_m^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell} \geq \frac{1}{512\cdot t(\lambda)\cdot(\lambda\cdot u(\lambda)\cdot q^*(\lambda))^2\cdot s_{4,\ell}(\lambda)}$$

are included in X_4^ℓ. Since this is true with probability $1-\exp\{-\lambda\}$ for each $m\in\mathfrak{L}$ and since $\mathfrak{L}$ is of polynomial size, we have that $\Pr[\mathsf{GOOD}]\geq 1-\mathsf{negl}(\lambda)$.

Now we claim

$$\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}}[W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}}|\mathsf{GOOD}] \leq \frac{1}{512\cdot t(\lambda)\cdot(\lambda\cdot u(\lambda)\cdot q^*(\lambda))^2}.$$

To see this, first by linearity of expectation we have

$$\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}}[W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}}\big|\mathsf{GOOD}] = \sum_{\substack{m:\\ m\notin X_4^\ell\\ m\in\mathfrak{L}}} \mathbb{E}[\alpha_m^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell}\big|\mathsf{GOOD}].$$

However, by definition of GOOD, when GOOD occurs, we have

$$m\in\mathfrak{L}\setminus X_4^\ell \implies \alpha_m^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell} \le \frac{1}{512\cdot t(\lambda)\cdot(\lambda\cdot u(\lambda)\cdot q^*(\lambda))^2\cdot s_{4,\ell}(\lambda)}$$

Since $|\mathfrak{L}| = s_{4,\ell}(\lambda)$, we obtain the desired upper bound on $\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}}[W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}}\big|\mathsf{GOOD}]$. Since $\Pr[\mathsf{GOOD}]\ge 1-\mathsf{negl}(\lambda)$, we get

$$\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}}[W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}}] \le \frac{1}{256\cdot t(\lambda)\cdot(\lambda\cdot u(\lambda)\cdot q^*(\lambda))^2}.$$

□

Now observe that Claim 5 follows immediately from Claim 6 and Claim 7 since $W^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell} = W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}} + W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}}$ for all $\mathcal{O}^*_{\mathsf{QKeyFireSign}},V^\ell,X_4^\ell,\mathfrak{L}$ (i.e. with probability 1). This completes the proof of Claim 5, and then the lemma follows as discussed above.

10.4 Proof of Theorem 20

In this section, we prove security in the last hybrid. That is, we prove Theorem 20, which says $\Pr[\mathsf{Hyb}_3 = 1]\le\mathsf{negl}(\lambda)$.

First we prove the following lemma.

Theorem 21. *Let $H:\{0,1\}^{v(\lambda)}\to\{0,1\}^\lambda$ be modeled as a quantum random oracle, and let $v(\lambda)$ be a superlogarithmic function.*

For any adversaries $(\mathcal{A}_1,\mathcal{A}_2)$, consider the following experiment.

$\mathsf{Exp}_{\mathcal{A}_1,\mathcal{A}_2}(\lambda)$

1. *Sample a random function $H:\{0,1\}^{v(\lambda)}\to\{0,1\}^\lambda$.*
2. *$\mathcal{A}_1$ is given quantum oracle access to H and outputs a message register R.*
3. *Sample $m^*\leftarrow\{0,1\}^{v(\lambda)}$.*
4. *$\mathcal{A}_2$ does not have access to H, and it receives (R,m^*) and outputs y^*.*
5. *Output 1 if $y^*=H(m^*)$, and 0 otherwise.*

Then, for any query-bounded $\mathcal{A}_1$,

$$\Pr\left[\mathsf{Exp}_{\mathcal{A}_1,\mathcal{A}_2}(\lambda)=1\right]\le\mathsf{negl}(\lambda).$$

Proof. Since m^* is unpredictable to $\mathcal{A}_1$, the result follows by Theorem 7. □

Observe that in Hyb_3, to win the game, the adversary needs to output a value sig^* such that $H_{\mathsf{sig}}(m^*) = sig^*$. Also observe that in this hybrid, the receiver adversary $\mathcal{B}_{2,u(\lambda)+1}$ does not have access to any oracles, and the sender adversary $\mathcal{B}_{1,u(\lambda)+1}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ makes polynomially many queries to the random oracle H_{sig} and to $\mathcal{O}_4^*$ (which can be simulated by querying H_{sig}). Thus the result follows by Theorem 21.

11 Proof of Lemma 7

In this section, we prove Lemma 7. Throughout this section, fix any $\ell^* \in [u(\lambda)+1]$. Therefore, we will prove $\left|\mathsf{Hyb}_{2,\ell^*,0} - \mathsf{Hyb}_{2,\ell^*,1}\right| \leq \frac{1}{\lambda \cdot u(\lambda) \cdot q^*(\lambda)}$. The proof will be done through a sequence of hybrids, each of which is constructed by modifying the previous one. We will denote our hybrids as Hyb'_i (*hybrid prime*), and their indices will be independent of the indices of the hybrids Hyb_j from Section 10.

We first define our hybrids.

$\mathsf{Hyb}'_{j,0}$ **for** $j \in [t(\lambda)]$**:**

$\mathsf{Hyb}'_{j,0}$

1. The challenger samples $sn^*, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QKeyFireSign.Spark}(1^\lambda)$.
2. Parse $\mathcal{O}^*_{\mathsf{QKeyFireSign}} = (\mathcal{O}^*_0, \mathcal{O}^*_1, \mathcal{O}^*_2, \mathcal{O}^*_{\mathsf{OSS.Sign}}, \mathcal{O}^*_3, \mathcal{O}^*_4) = sn^*$.

   Round-collapsed Simulation for First $\ell^* - 1$ Rounds:
3. Run $\mathcal{B}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,\ell^*-1}(\mathsf{R}_{\mathsf{flame}})$ to get $\mathsf{R}^{\ell^*-1}_{1,\mathsf{internal}}$ and $(T^{\ell^*-1}, S^{\ell^*-1})$.

   Round ℓ^*:
4. $\mathcal{A}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,\ell^*}$ receives $\mathsf{R}^{\ell^*-1}_{1,\mathsf{internal}}$ and S^{ℓ^*-1}, and it outputs an internal state register $\mathsf{R}^{\ell^*}_{1,\mathsf{internal}}$ and a leakage message L^{ℓ^*}
5. $\mathcal{C}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,\ell^*,j,0}$ receives T^{ℓ^*-1}, L^{ℓ^*} and produces a message $Y^{\ell^*,j,0}$.
6. Run $\mathcal{C}^{(\mathcal{O}^*_{\mathsf{OSS.Sign}}, \mathcal{O}^*_3, \mathcal{O}^*_4)}_{2,\ell^*,j,0}(Y^{\ell^*,j,0})$ to obtain $(\mathsf{R}^{\ell^*}_{2,\mathsf{internal}}, S^{\ell^*})$

   Original Protocol Continuation:
7. For $i \in \{\ell^*+1, \ldots, u(\lambda)+1\}$:
   1. $\mathcal{A}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,i}$ receives $\mathsf{R}^{i-1}_{1,\mathsf{internal}}$ and S^{i-1}, and it outputs a leakage message L^i and an internal state register $\mathsf{R}^i_{1,\mathsf{internal}}$.
   2. $\mathcal{A}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{2,i}$ receives $\mathsf{R}^{i-1}_{2,\mathsf{internal}}$ and L^i, and it outputs a message S^i and an internal state register $\mathsf{R}^i_{2,\mathsf{internal}}$.
8. Parse $(m^*, sig^*) = S^{u(\lambda)+1}$.
9. Output $\mathcal{O}^*_4(m^*, sig^*)$.

$\mathsf{Hyb}'_{j,1}$ **for** $j \in [t(\lambda)]$**:**

$\mathsf{Hyb}'_{j,1}$

1. The challenger samples $sn^*, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QKeyFireSign.Spark}(1^\lambda)$.

2. Parse $\mathcal{O}^*_{\mathsf{QKeyFireSign}} = (\mathcal{O}^*_0, \mathcal{O}^*_1, \mathcal{O}^*_2, \mathcal{O}^*_{\mathsf{OSS.Sign}}, \mathcal{O}^*_3, \mathcal{O}^*_4) = sn^*$.

Round-collapsed Simulation for First $\ell^* - 1$ Rounds:

3. Run $\mathcal{B}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,\ell^*-1}(\mathsf{R}_{\mathsf{flame}})$ to get $\mathsf{R}^{\ell^*-1}_{1,\mathsf{internal}}$ and $(T^{\ell^*-1}, S^{\ell^*-1})$.

Round ℓ^*:

4. $\mathcal{A}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,\ell^*}$ receives $\mathsf{R}^{\ell^*-1}_{1,\mathsf{internal}}$ and S^{ℓ^*-1}, and it outputs an internal state register $\mathsf{R}^{\ell^*}_{1,\mathsf{internal}}$ and a leakage message L^{ℓ^*}

5. $\mathcal{C}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,\ell^*,j,1}$ receives T^{ℓ^*-1}, L^{ℓ^*} and produces a message $Y^{\ell^*,j,1}$.

6. Run $\mathcal{C}^{(\mathcal{O}^*_{\mathsf{OSS.Sign}}, \mathcal{O}^*_3, \mathcal{O}^*_4)}_{2,\ell^*,j,1}(Y^{\ell^*,j,1})$ to obtain $(\mathsf{R}^{\ell^*}_{2,\mathsf{internal}}, S^{\ell^*})$

Original Protocol Continuation:

7. For $i \in \{\ell^* + 1, \ldots, u(\lambda) + 1\}$:
    1. $\mathcal{A}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,i}$ receives $\mathsf{R}^{i-1}_{1,\mathsf{internal}}$ and S^{i-1}, and it outputs a leakage message L^i and an internal state register $\mathsf{R}^{i}_{1,\mathsf{internal}}$.
    2. $\mathcal{A}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{2,i}$ receives $\mathsf{R}^{i-1}_{2,\mathsf{internal}}$ and L^i, and it outputs a message S^i and an internal state register $\mathsf{R}^{i}_{2,\mathsf{internal}}$.

8. Parse $(m^*, sig^*) = S^{u(\lambda)+1}$.

9. Output $\mathcal{O}^*_4(m^*, sig^*)$.

$\mathsf{Hyb}'_{j,2}$ **for** $i \in \{0, \ldots, t(\lambda)\}$**:**

$\mathsf{Hyb}'_{j,2}$

1. The challenger samples $sn^*, \mathsf{R}_{\mathsf{flame}} \leftarrow \mathsf{QKeyFireSign.Spark}(1^\lambda)$.

2. Parse $\mathcal{O}^*_{\mathsf{QKeyFireSign}} = (\mathcal{O}^*_0, \mathcal{O}^*_1, \mathcal{O}^*_2, \mathcal{O}^*_{\mathsf{OSS.Sign}}, \mathcal{O}^*_3, \mathcal{O}^*_4) = sn^*$.

Round-collapsed Simulation for First $\ell^* - 1$ Rounds:

3. Run $\mathcal{B}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,\ell^*-1}(\mathsf{R}_{\mathsf{flame}})$ to get $\mathsf{R}^{\ell^*-1}_{1,\mathsf{internal}}$ and $(T^{\ell^*-1}, S^{\ell^*-1})$.

Round ℓ^*:

4. $\mathcal{A}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,\ell^*}$ receives $\mathsf{R}^{\ell^*-1}_{1,\mathsf{internal}}$ and S^{ℓ^*-1}, and it outputs an internal state register $\mathsf{R}^{\ell^*}_{1,\mathsf{internal}}$ and a leakage message L^{ℓ^*}

5. $\mathcal{C}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}_{1,\ell^*,j,2}$ receives T^{ℓ^*-1}, L^{ℓ^*} and produces a message $Y^{\ell^*,j,2}$.

6. Run $\mathcal{C}^{(\mathcal{O}^*_{\mathsf{OSS.Sign}}, \mathcal{O}^*_3, \mathcal{O}^*_4)}_{2,\ell^*,j,2}(Y^{\ell^*,j,2})$ to obtain $(\mathsf{R}^{\ell^*}_{2,\mathsf{internal}}, S^{\ell^*})$

Original Protocol Continuation:

7. For $i \in \{\ell^* + 1, \ldots, u(\lambda) + 1\}$:
   1. $\mathcal{A}_{1,i}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $\mathsf{R}^{i-1}_{1,\mathsf{internal}}$ and S^{i-1}, and it outputs a leakage message L^i and an internal state register $\mathsf{R}^{i}_{1,\mathsf{internal}}$.
   2. $\mathcal{A}_{2,i}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}}$ receives $\mathsf{R}^{i-1}_{2,\mathsf{internal}}$ and L^i, and it outputs a message S^i and an internal state register $\mathsf{R}^{i}_{2,\mathsf{internal}}$.
8. Parse $(m^*, sig^*) = S^{u(\lambda)+1}$.
9. Output $\mathcal{O}^*_4(m^*, sig^*)$.

Lemma 13. $\mathsf{Hyb}'_{0,2} \equiv \mathsf{Hyb}_{2,\ell^*,0}$

Proof. This follows by the definition of the adversaries $\mathcal{C}_{1,\ell,0,2}$,$\mathcal{C}_{2,\ell,0,2}$. □

Lemma 14. $\mathsf{Hyb}'_{t(\lambda),2} \equiv \mathsf{Hyb}_{2,\ell^*,1}$.

Proof. Follows by the definition of the adversary $\mathcal{C}_{1,\ell}, \mathcal{C}_{2,\ell}$ which are defined to be $\mathcal{C}_{1,\ell,t(\lambda),2}$,$\mathcal{C}_{2,\ell,t(\lambda),2}$. □

Lemma 15. *For* $i \in [t(\lambda)]$, $\left|\mathsf{Hyb}'_{i-1,2} - \mathsf{Hyb}'_{i,0}\right| \leq \frac{1}{\lambda^2 \cdot u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda)}$ *and* $\left|\mathsf{Hyb}'_{i,0} - \mathsf{Hyb}'_{i,1}\right| \leq \frac{1}{\lambda^2 \cdot u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda)}$.

We prove this lemma in Section 11.1.

Lemma 16. *For* $i \in [t(\lambda)]$, $\left|\mathsf{Hyb}'_{i,1} - \mathsf{Hyb}'_{i,2}\right| \leq \frac{1}{\lambda^2 \cdot u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda)}$

We prove this lemma in Section 11.2.

Finally, Lemma 7 follows by simply combining these lemmata.

11.1 Proof of Lemma 15

In this section, we prove Lemma 15. We will only explicitly prove the first part, and the second part follows similarly.

Fix any $i^* \in \{1, \ldots, t(\lambda)\}$. Thus, we will prove $\left|\mathsf{Hyb}'_{i^*-1,2} - \mathsf{Hyb}'_{i^*,0}\right| \leq \frac{1}{\lambda^2 \cdot u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda)}$. Observe that the difference between the two hybrids is that in $\mathsf{Hyb}'_{i^*-1,2}$ we run $\mathcal{C}_{2,\ell^*,i^*-1,2}$ directly, whereas in $\mathsf{Hyb}'_{i^*,0}$ we instead run $\mathcal{C}_{2,\ell^*,i^*,0}$ which simulates $\mathcal{C}_{2,\ell^*,i^*-1,2}$ while using an estimated oracle for its first query to $\mathcal{O}^*_0$. We will use the strong incompressibility property of OSS to show that this only makes a difference of $\frac{1}{\lambda^2 \cdot u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda)}$ in statistical distance. In this proof, whenever we refer to incompressibility, we will be referring to the strong incompressibility of OSS. For the strong incompressibility property of OSS, consider the following adversaries $\mathcal{P}_1, \mathcal{P}_2$. $\mathcal{P}_1$ simulates $\mathsf{Hyb}'_{i^*-1,2}$ until $Y^{\ell^*,i^*,1}$ is produced, and outputs $Y^{\ell^*,i^*,1}$. Then, $\mathcal{P}_2$ receives this value and runs $\mathcal{C}_{\ell^*,2,i^*-1,2}$ with it until right after first query to $\mathcal{O}^*_0$. When considering $\mathcal{P}_1, \mathcal{P}_2$ for incompressibility of OSS, we assume that $\mathcal{P}_1, \mathcal{P}_2$ are only oracle-adversaries for OSS, and the other oracles are just simulated, e.g. they sample $H_0, H_1, H_{\mathsf{sig}}$ in advance and both use this sample to simulate the oracle calls to relevant oracles.

Now let $\mathfrak{L}$ denote the strong incompressibility list for these adversaries, and we know $|\mathfrak{L}| = s_{OSS,\ell^*}(\lambda) = 2048 \cdot \lambda \cdot (t^2(\lambda) \cdot \ell^*) \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2 \cdot (\ell^* - 1) + 2 \cdot i^*$ since $\mathcal{P}_1$ makes at most

$\frac{s_{OSS,\ell^*}(\lambda)}{2}$ calls to $\mathcal{O}^*_{\mathsf{OSS.GenQKey}}$. The list depends on the oracle itself and the *leakage* $Y^{\ell^*,i^*,1}$, but we will not write this dependence explicitly for conciseness.

Let $\gamma^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2}}_{ivk,isig,z}$ denote the query amplitude of $\mathcal{C}_{\ell^*,2,i^*-1,2}(Y^{\ell^*,i^*-1,2})$ on input $ivk, isig, z$ during its first query to $\mathcal{O}^*_0$. We also define the following. Here, $X^{\ell^*}_{0,i^*}$ is the list sampled during execution of $\mathcal{C}_{1,\ell^*,i^*,0}$.

$$\alpha^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2}}_{ivk,isig} = \begin{cases} |\gamma^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2}}_{ivk,isig}|^2, \text{ if } \mathcal{O}^*_{\mathsf{OSS.Ver}}(ivk,0,isig)=1 \\ 0, \text{ if } \mathcal{O}^*_{\mathsf{OSS.Ver}}(ivk,0,isig)=0 \end{cases}$$

$$W^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2},X^{\ell^*}_{0,i^*}} = \sum_{\substack{ivk,isig:\\(ivk,isig)\notin X^{\ell^*}_{0,i^*}}} \alpha^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2}}_{ivk,isig}$$

$$W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2},X^{\ell^*}_{0,i^*},\mathfrak{L}} = \sum_{\substack{ivk,isig:\\(ivk,isig)\notin X^{\ell^*}_{0,i^*}\\(ivk,isig)\notin \mathfrak{L}}} \alpha^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2}}_{ivk,isig}$$

$$W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2},X^{\ell^*}_{0,i^*},\mathfrak{L}} = \sum_{\substack{ivk,isig:\\(ivk,isig)\notin X^{\ell^*}_{0,i^*}\\(ivk,isig)\in \mathfrak{L}}} \alpha^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2}}_{ivk,isig}$$

Now we make the following claims.

Claim 8.

$$\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2},X^{\ell^*}_{0,i^*},\mathfrak{L}}[W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2},X^{\ell^*}_{0,i^*},\mathfrak{L}}] \leq \mathsf{negl}(\lambda).$$

Proof. Observe that with probability 1, we have

$$W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2},X^{\ell^*}_{0,i^*},\mathfrak{L}} \leq \sum_{\substack{ivk,isig:\\(ivk,isig)\notin \mathfrak{L}}} \alpha^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2}}_{ivk,isig}$$

However, expectation of the value on the right is negligible by incompressibility of OSS (with the adversaries $\mathcal{P}_1, \mathcal{P}_2$). □

Claim 9. *For all sufficiently large $\lambda > 0$.*

$$\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2},X^{\ell^*}_{0,i^*},\mathfrak{L}}[W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2},X^{\ell^*}_{0,i^*},\mathfrak{L}}] \leq \frac{1}{256\cdot\lambda^4\cdot(u(\lambda)\cdot q^*(\lambda)\cdot t(\lambda))^2}$$

Proof. First we claim that, for all $\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*,i^*-1,2}$, with overwhelming probability all $ivk, isig$ that satisfies

$$\alpha^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*-1,2}}_{ivk,isig} \geq \frac{1}{512\cdot s_{OSS,\ell^*}(\lambda)\cdot\lambda^4\cdot(u(\lambda)\cdot q^*(\lambda)\cdot t(\lambda))^2}$$

is contained in $X^{\ell^*}_{0,i^*}$. To see this, observe that $X^{\ell^*}_{0,i^*}$ is created by measuring the first query of $\mathcal{C}_{\ell^*,2,i^*-1,2}$ to $\mathcal{O}_0$ for $r_{OSS,\ell^*}(\lambda) = 4194304\cdot\lambda^{10}\cdot t^2(\lambda)\cdot(\ell^*)^5\cdot(u(\lambda)\cdot q^*(\lambda)\cdot t(\lambda))^4$ times. Thus, for

any $ivk, isig$, the probability that $(ivk, isig) \notin X^{\ell^*}_{0,i^*}$ is at most $\exp\{-r_{OSS,\ell^*}(\lambda) \cdot \alpha_{ivk,isig}\}$. Since we have

$$1 - \exp\{-r_{OSS,\ell^*}(\lambda) \cdot \frac{1}{512 \cdot s_{OSS,\ell^*}(\lambda) \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}\} = 1 - \exp\{-\lambda\}$$

the claim follows. Now define GOOD to be the event that all $(ivk, isig) \in \mathfrak{L}$ that satisfies

$$\alpha^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}}_{ivk,isig} \geq \frac{1}{512 \cdot s_{OSS,\ell^*}(\lambda) \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$

are included in $X^{\ell^*}_{0,i^*}$. Since this is true with probability $1 - \exp\{-\lambda\}$ for each $ivk, isig \in \mathfrak{L}$ individually, and since $|\mathfrak{L}|$ is polynomial, we have that $\Pr[\mathsf{GOOD}] \geq 1 - \mathsf{negl}(\lambda)$.

Now we claim

$$\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}, X^{\ell^*}_{0,i^*}, \mathfrak{L}}[W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}, X^{\ell^*}_{0,i^*}, \mathfrak{L}} \big| \mathsf{GOOD}] \leq \frac{1}{512 \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$

To see this, first by linearity of expectation we have

$$\begin{aligned}
&\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}, X^{\ell^*}_{0,i^*}, \mathfrak{L}}[W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}, X^{\ell^*}_{0,i^*}, \mathfrak{L}} \big| \mathsf{GOOD}] \\
&= \sum_{\substack{ivk,isig: \\ (ivk,isig) \notin X^{\ell^*}_{0,i^*} \\ (ivk,isig) \in \mathfrak{L}}} \mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}} \left[\alpha^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}}_{ivk,isig} \big| \mathsf{GOOD} \right].
\end{aligned}$$

However, by definition of GOOD, when GOOD occurs, we have

$$ivk, isig \in \mathfrak{L} \setminus X^{\ell^*}_{0,i^*} \implies \alpha^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}}_{ivk,isig} \leq \frac{1}{512 \cdot s_{OSS,\ell^*}(\lambda) \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$

Since $|\mathfrak{L}| = s_{OSS,\ell^*}(\lambda)$, we obtain the desired upper bound on

$$\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}, X^{\ell^*}_{0,i^*}, \mathfrak{L}}[W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}, X^{\ell^*}_{0,i^*}, \mathfrak{L}} \big| \mathsf{GOOD}].$$

Since $\Pr[\mathsf{GOOD}] \geq 1 - \mathsf{negl}(\lambda)$, we get

$$\mathbb{E}_{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}, X^{\ell^*}_{0,i^*}, \mathfrak{L}}[W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}, X^{\ell^*}_{0,i^*}, \mathfrak{L}}] \leq \frac{1}{256 \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$

□

Now, observe that Lemma 15 follows from these two lemmata. To see this, observe that $W^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}, X^{\ell^*}_{0,i^*}}$ is an upper bound on the total weight on inputs at which the two oracles might differ. Since we have $W^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}, X^{\ell^*}_{0,i^*}} = W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}, X^{\ell^*}_{0,i^*}, \mathfrak{L}} + W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*, i^*-1,2}, X^{\ell^*}_{0,i^*} \mathfrak{L}}$ with probability 1, the result follows by the linearity of expectation, Theorem 6 and Jensen's inequality.

11.2 Proof of Lemma 16

Fix any $i^* \in \{1, \ldots, t(\lambda)\}$. Thus, we will prove $\left|\mathsf{Hyb}'_{i^*,1} - \mathsf{Hyb}'_{i^*,2}\right| \leq \frac{1}{\lambda^2 \cdot u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda)}$. The proof will follow similarly to Lemma 9.

Observe that the difference between the two hybrids is that in $\mathsf{Hyb}'_{i^*,1}$ we run $\mathcal{C}_{2,\ell^*,i^*,1}$ directly, whereas in $\mathsf{Hyb}'_{i^*,2}$ we instead run $\mathcal{C}_{2,\ell^*,i^*,2}$ which simulates $\mathcal{C}_{2,\ell^*,i^*,1}$ while using an estimated oracle for its first query to $\mathcal{O}_2^*$. We will use the incompressibility property of H_0, H_1 to show that this only makes a difference of $\frac{1}{\lambda^2 \cdot u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda)}$ in statistical distance.

Let $\gamma_{ivk,y_0,y_1,z}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*,i^*,1}}$ denote the query amplitude of $\mathcal{C}_{\ell^*,2,i^*,1}(Y^{\ell^*,i^*,1})$ on input ivk, y_0, y_1, z during its first query to $\mathcal{O}_2^*$. We also define the following.

$$\alpha_{ivk,z}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*,i^*,1}} = \sum_{y_0,y_1 : \mathcal{O}_2^*(ivk,y_0,y_1,z) \neq \perp} |\gamma_{ivk,y_0,y_1,z}^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*,i^*,1}}|^2$$

Now let $(X^{\ell'}_{0,j'})_{\ell' \in [\ell^*], j' \in [i^*]}$, $(X^{\ell'}_{1,j'})_{\ell' \in [\ell^*], j' \in [i^*]}$, $(X^{\ell'}_{2,j'})_{\ell' \in [\ell^*], j' \in [i^*]-1}$, $(D^{\ell'}_{0,j'})_{\ell' \in [\ell^*], j' \in [i^*]}$, $(D^{\ell'}_{1,j'})_{\ell' \in [\ell^*], j' \in [i^*]}$, $(D^{\ell'}_{2,j'})_{\ell' \in [\ell^*], j' \in [i^*]-1}$ be the estimated lists and databases in $Y^{\ell^*,i^*,1}$ when it is parsed and fully unwrapped. With abuse of notation, sometimes we will refer to the projection of the sets $X^{\ell'}_{\cdot,j'}$ onto their elements' (which are tuples) subcomponents also as $X^{\ell'}_{\cdot,j'}$ (e.g. we will write $X^{\ell'}_{2,j'}$ to mean the set of pairs (ivk, z) obtained by projecting each tuple in the actual set $X^{\ell'}_{2,j'}$ to its first and fourth elements) - however, the meaning will clear from context. Now we claim the following.

Claim 10.

$$\Pr\left[\left(\bigcup_{\ell' \in [\ell^*], j' \in [i^*]} X^{\ell'}_{0,j'}\right) \bigcap \left(\bigcup_{\ell' \in [\ell^*], j' \in [i^*]} X^{\ell'}_{1,j'}\right) = \emptyset\right] \geq 1 - \mathsf{negl}(\lambda).$$

Proof. Observe that if this intersection is not empty, then there is ivk such that $ivk \in X^{\ell'}_{0,j'}$ and $ivk \in X^{\ell''}_{1,j''}$ for some ℓ', ℓ'', j', j''. However, by construction we know that this implies that we can obtain $isig_0, isig_1$ from $Y^{\ell^*,i^*,1}$ such that $\mathcal{O}_{\mathsf{OSS.Ver}^*}(ivk, 0, isig_0) = \mathcal{O}_{\mathsf{OSS.Ver}^*}(ivk, 1, isig_1) = 1$. Since $Y^{\ell^*,i^*,1}$ is obtained by making polynomially many queries to the oracles of OSS, the claim follows by the one-shot security of OSS. □

Now we define the following (some dependencies are implicit to keep the notation concise).

$$W_0^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*,i^*,1}, X^{\ell^*}_{2,i^*}} = \sum_{\substack{ivk,z: \\ ivk \notin \left(\bigcup_{\ell' \in [\ell^*], j' \in [i^*]} X^{\ell'}_{0,j'}\right) \\ (ivk,z) \notin X^{\ell^*}_{2,i^*}}} \alpha_{ivk,z}$$

$$W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*,i^*,1}, X^{\ell^*}_{2,i^*}} = \sum_{\substack{ivk,z: \\ ivk \notin \left(\bigcup_{\ell' \in [\ell^*], j' \in [i^*]} X^{\ell'}_{1,j'}\right) \\ (ivk,z) \notin X^{\ell^*}_{2,i^*}}} \alpha_{ivk,z}$$

$$W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*,i^*,1}, X^{\ell^*}_{2,i^*}} = \sum_{\substack{ivk,z: \\ ivk \in \left(\bigcup_{\ell' \in [\ell^*], j' \in [i^*]} X^{\ell'}_{0,j'}\right) \bigcap \left(\bigcup_{\ell' \in [\ell^*], j' \in [i^*]} X^{\ell'}_{1,j'}\right)}} \alpha_{ivk,z}$$

Now we make the following claims.

Claim 11.

$$\mathbb{E}[W_2^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*,1},X^{\ell^*}_{2,i^*}}] \leq \mathsf{negl}(\lambda). \tag{2}$$

Proof. This simply follows from the definition of W_2 and Claim 10. □

Claim 12. *For all sufficiently large $\lambda > 0$.*

$$\mathbb{E}[W_0^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*,1},X^{\ell^*}_{2,i^*}}] \leq \frac{1}{128 \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$
$$\mathbb{E}[W_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*,1},X^{\ell^*}_{2,i^*}}] \leq \frac{1}{128 \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$

Proof. We will only prove the first one (using incompressibility of the random oracle H_0), and the second bound follows by the same argument (but this time relying on the incompressibility of H_1).

Now we will use the incompressibility property (Theorem 15) of the random oracle H_0 with the predicate P defined as $P(ivk||z) = 1$ if $ivk \in S$, where $S = \left(\bigcup_{\ell' \in [\ell^*], j' \in [i^*]} X^{\ell'}_{0,j'}\right)$. Note that this set S is exactly the set of prefixes on which our adversary has *section queries* (otherwise it only makes normal queries and verification queries). In more detail, let $\mathcal{P}_1$ be the adversary that simulates $\mathsf{Hyb}'_{i^*,1}$ until the point where $\mathcal{C}_{1,\ell^*,i^*,2}$ outputs its message, and it also outputs S as its predicate. Then $\mathcal{P}_2$ simulates $\mathcal{C}_{2,\ell^*,i^*,1}$ until right after its first query to $\mathcal{O}^*_2$. We invoke Theorem 15 for this pair of adversaries, and let $\mathfrak{L}$ denote the incompressibility list that is promised to exist, and we know $|\mathfrak{L}| = \ell^* \cdot t(\lambda)$ since $\mathcal{P}_1$ makes at most $\ell^* \cdot t(\lambda)$ normal queries. We then split W_0 into two summands U_0, U_1:

$$U_0^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*,1},X^{\ell^*}_{2,i^*}} = \sum_{\substack{ivk,z: \\ ivk \notin \left(\bigcup_{\ell' \in [\ell^*], j' \in [i^*]} X^{\ell'}_{0,j'}\right) \\ (ivk,z) \notin X^{\ell^*}_{2,i^*} \\ (ivk,z) \notin \mathfrak{L}}} \alpha_{ivk,z}$$

$$U_1^{\mathcal{O}^*_{\mathsf{QKeyFireSign}},Y^{\ell^*,i^*,1},X^{\ell^*}_{2,i^*}} = \sum_{\substack{ivk,z: \\ ivk \notin \left(\bigcup_{\ell' \in [\ell^*], j' \in [i^*]} X^{\ell'}_{0,j'}\right) \\ (ivk,z) \notin X^{\ell^*}_{2,i^*} \\ (ivk,z) \in \mathfrak{L}}} \alpha_{ivk,z}$$

It is easy to see that $\mathbb{E}[U_0] \leq \mathsf{negl}(\lambda)$ by the incompressibility of H_0, thus, we will focus on bounding U_1.

First we claim that, for all $\mathcal{O}^*_{\mathsf{QKeyFireSign}}, Y^{\ell^*,i^*,1}$, with overwhelming probability we have that any ivk, z that satisfies

$$\alpha_{ivk,z} \geq \frac{1}{512 \cdot (t(\lambda) \cdot \ell^*) \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$

is contained in $X^{\ell^*}_{2,i^*}$. To see this, observe that $X^{\ell^*}_{2,i^*}$ is created by measuring the first query of $\mathcal{C}_{\ell^*,2,i^*,1}$ to $\mathcal{O}_2$ for $512 \cdot \lambda \cdot (t(\lambda) \cdot \ell^*) \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2$ times. Thus, for any ivk, z such that

$\alpha_{ivk,z}$ satisfies the above condition, the probability that $(ivk, z) \notin X^{\ell^*}_{2,i^*}$ is at most $\exp\{-\lambda\}$. Now define GOOD to be the event that all $(ivk, z) \in \mathfrak{L}$ that satisfies

$$\alpha_{ivk,z} \geq \frac{1}{512 \cdot (t(\lambda) \cdot \ell^*) \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$

are included in $X^{\ell^*}_{2,i^*}$. Since this is true with probability $1-\exp\{-\lambda\}$ for each $ivk, z \in \mathfrak{L}$ individually, and since $|\mathfrak{L}|$ is polynomial, we have that $\Pr[\mathsf{GOOD}] \geq 1 - \mathsf{negl}(\lambda)$.

Now we claim

$$\mathbb{E}[U_1 \big| \mathsf{GOOD}] \leq \frac{1}{512 \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$

To see this, first by linearity of expectation we have

$$\mathbb{E}[U_1 \big| \mathsf{GOOD}] = \sum_{\substack{ivk,z: \\ ivk \notin \left(\bigcup_{\ell' \in [\ell^*], j' \in [i^*]} X^{\ell'}_{0,j'}\right) \\ (ivk,z) \notin X^{\ell^*}_{2,i^*} \\ (ivk,z) \in \mathfrak{L}}} \mathbb{E}[\alpha_{ivk,z} | \mathsf{GOOD}]$$

However, by definition of GOOD, when GOOD occurs, we have

$$ivk, z \in \mathfrak{L} \setminus X^{\ell^*}_{2,i^*} \implies \alpha_{ivk,z} \leq \frac{1}{512 \cdot (t(\lambda) \cdot \ell^*) \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$

Since $|\mathfrak{L}| = \ell^* \cdot t(\lambda)$, we obtain the desired upper bound on expectation conditioned on GOOD. Since $\Pr[\mathsf{GOOD}] \geq 1 - \mathsf{negl}(\lambda)$, we also get

$$\mathbb{E}[U_1] \leq \frac{1}{256 \cdot \lambda^4 \cdot (u(\lambda) \cdot q^*(\lambda) \cdot t(\lambda))^2}$$

which completes the proof. □

Observe that Lemma 16 follows from the above claims. To see this, observe that $W_0 + W_1 + W_2$ is an upper bound on the total weight on inputs at which the two different oracles in the hybrids might differ. Thus the result follows by the linearity of expectation, Theorem 6 and Jensen's inequality.

References

[Aar09] Scott Aaronson. Quantum copy-protection and quantum money. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 229–242. IEEE, 2009.

[AARR03] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM side—channel(s). In Burton S. Kaliski, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 29–45, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[AC12] Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. In Howard J. Karloff and Toniann Pitassi, editors, *44th Annual ACM Symposium on Theory of Computing*, pages 41–60, New York, NY, USA, May 19–22, 2012. ACM Press.

[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 Annual ACM Symposium on Theory of Computing*, pages 255–268, Chicago, IL, USA, June 22–26, 2020. ACM Press.

[AK25] Avantika Agarwal and Srijita Kundu. A cautionary note on quantum oracles, 2025.

[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, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 526–555, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.

[BBBV97] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523, 1997.

[BBJ16] Ning Bao, Adam Bouland, and Stephen P. Jordan. Grover search and the no-signaling principle. *Phys. Rev. Lett.*, 117:120501, Sep 2016.

[BGMS25] James Bartusek, Aparna Gupte, Saachi Mutreja, and Omri Shmueli. Classical obfuscation of quantum circuits via publicly-verifiable qfhe, 2025.

[BNZ25] John Bostanci, Barak Nehoran, and Mark Zhandry. A general quantum duality for representations of groups with applications to quantum money, lightning, and fire. In Michal Koucký and Nikhil Bansal, editors, *57th Annual ACM Symposium on Theory of Computing*, pages 201–212, Prague, Czechia, June 23–27, 2025. ACM Press.

[ÇG24] Alper Çakan and Vipul Goyal. Unbounded leakage-resilient encryption and signatures. Cryptology ePrint Archive, Paper 2024/1876, 2024.

[ÇG25] Alper Çakan and Vipul Goyal. How to copy-protect all puncturable functionalities without conjectures: A unified solution to quantum protection. Cryptology ePrint Archive, Paper 2025/1197, 2025.

[ÇGLZR24] Alper Çakan, Vipul Goyal, Chen-Da Liu-Zhang, and João Ribeiro. Unbounded leakage-resilience and intrusion-detection in a quantum world. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024: 22nd Theory of Cryptography Conference, Part II*, volume 15365 of *Lecture Notes in Computer Science*, pages 159–191, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland.

[CLLZ21] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. Hidden cosets and applications to unclonable cryptography. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 556–584, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.

[DS23] Marcel Dall'Agnol and Nicholas Spooner. On the necessity of collapsing for post-quantum and quantum commitments. 2023.

[FGLE12] Steven T Flammia, David Gross, Yi-Kai Liu, and Jens Eisert. Quantum tomography via compressed sensing: error bounds, sample complexity and efficient estimators. *New Journal of Physics*, 14(9):095022, 2012.

[FGSY25] Bill Fefferman, Soumik Ghosh, Makrand Sinha, and Henry Yuen. The hardness of learning quantum circuits and its cryptographic applications, 2025.

[GMMY24] Eli Goldin, Tomoyuki Morimae, Saachi Mutreja, and Takashi Yamakawa. CountCrypt: Quantum cryptography between QCMA and PP. Cryptology ePrint Archive, Paper 2024/1707, 2024.

[Gol01] Oded Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.

[Gol04] Oded Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2004.

[HV25] Andrew Huang and Vinod Vaikuntanathan. A simple and efficient one-shot signature scheme, 2025.

[KMNY24] Fuyuki Kitagawa, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Quantum public-key encryption with tamper-resilient public keys from one-way functions. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part VII*, volume 14926 of *Lecture Notes in Computer Science*, pages 93–125, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, pages 104–113, 1996.

[LLQZ22] Jiahui Liu, Qipeng Liu, Luowen Qian, and Mark Zhandry. Collusion resistant copy-protection for watermarkable functionalities. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022: 20th Theory of Cryptography Conference, Part I*, volume 13747 of *Lecture Notes in Computer Science*, pages 294–323, Chicago, IL, USA, November 7–10, 2022. Springer, Cham, Switzerland.

[MY22] Tomoyuki Morimae and Takashi Yamakawa. Quantum commitments and signatures without one-way functions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part I*, volume 13507 of *Lecture Notes in*

Computer Science, pages 269–295, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland.

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[NZ24] Barak Nehoran and Mark Zhandry. A computational separation between quantum no-cloning and no-telegraphing. In Venkatesan Guruswami, editor, *ITCS 2024: 15th Innovations in Theoretical Computer Science Conference*, volume 287, pages 82:1–82:23, Berkeley, CA, USA, January 30 – February 2, 2024. Leibniz International Proceedings in Informatics (LIPIcs).

[OW16] Ryan O'Donnell and John Wright. Efficient quantum tomography. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 899–912, 2016.

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and counter-measures for smart cards. In Isabelle Attali and Thomas Jensen, editors, *Smart Card Programming and Security*, pages 200–210, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[Shm22] Omri Shmueli. Semi-quantum tokenized signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 296–319, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

[SZ25] Omri Shmueli and Mark Zhandry. On one-shot signatures, quantum vs. classical binding, and obfuscating permutations. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part II*, volume 16001 of *Lecture Notes in Computer Science*, pages 350–383, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.

[Wie83] Stephen Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, jan 1983.

[Win99] A. Winter. Coding theorem and strong converse for quantum channels. *IEEE Transactions on Information Theory*, 45(7):2481–2485, 1999.

[YZ22] Takashi Yamakawa and Mark Zhandry. Verifiable quantum advantage without structure. In *63rd Annual Symposium on Foundations of Computer Science*, pages 69–74, Denver, CO, USA, October 31 – November 3, 2022. IEEE Computer Society Press.

[Zha12] Mark Zhandry. How to construct quantum random functions. In *53rd Annual Symposium on Foundations of Computer Science*, pages 679–687, New Brunswick, NJ, USA, October 20–23, 2012. IEEE Computer Society Press.

[Zha19] Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in*

Cryptology – CRYPTO 2019, Part II, volume 11693 of *Lecture Notes in Computer Science*, pages 239–268, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.

[Zha25] Mark Zhandry. How to model unitary oracles. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part II*, volume 16001 of *Lecture Notes in Computer Science*, pages 237–268, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.

Appendix

A Alternative Equivalent Definitions for Quantum Protection Notions

We recall the definition of LOCC leakage-resilience for the case of signing and decryption. Note that this is a special case of the general definition (Definition 8), and we write the specialized version explicitly for ease of exposition.

Definition 25 (Signature Scheme with LOCC Leakage-Resilience [ÇGLZR24, ÇG24])**.** *Consider the following game between the challenger and an adversary $\mathcal{A}$.*

$\underline{\mathsf{LR}-\mathsf{SIGN}-\mathsf{GAME}_{\mathcal{A}}(1^\lambda)}$

1. *The challenger samples $vk, \mathsf{R}_{\mathsf{key}} \leftarrow \mathsf{SIG.Setup}(1^\lambda)$ and submits vk to $\mathcal{A}$.*
2. *The adversary obtains leakage on $\mathsf{R}_{\mathsf{key}}$ adaptively over any number of rounds as in Definition 8.*
3. *The challenger samples $m^* \leftarrow \mathcal{M}$ and submits m^* to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs a forged signature sig.*
5. *The challenger outputs 1 if and only if $\mathsf{SIG.Ver}(vk, m^*, sig)$.*

A digital signature scheme SIG *is said to satisfy* LOCC leakage-resilience *if for any QPT adversary $\mathcal{A}$,*

$$\Pr\left[\mathsf{LR}-\mathsf{SIGN}-\mathsf{GAME}_{\mathcal{A}}(1^\lambda)=1\right] \leq \frac{1}{|\mathcal{M}|} + \mathsf{negl}(\lambda).$$

Definition 26 (Public-Key Encryption with LOCC Leakage-Resilience [ÇGLZR24, ÇG24])**.** *Consider the following game between the challenger and an adversary $\mathcal{A}$.*

$\underline{\mathsf{LR}-\mathsf{PKE}-\mathsf{GAME}_{\mathcal{A}}(1^\lambda)}$

1. *The challenger samples $pk, \mathsf{R}_{\mathsf{key}} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ and submits pk to $\mathcal{A}$.*
2. *The adversary obtains leakage on $\mathsf{R}_{\mathsf{key}}$ adaptively over any number of rounds as in Definition 8.*
3. *The adversary $\mathcal{A}$ outputs two messages m_0, m_1.*
4. *The challenger samples $b \leftarrow \{0,1\}$ and $ct \leftarrow \mathsf{PKE.Enc}(pk, m_b)$ and submits ct to $\mathcal{A}$.*
5. *$\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.*
6. *The challenger outputs 1 if and only if $b' = b$.*

A public-key encryption scheme PKE *is said to satisfy* LOCC leakage-resilience *if for any QPT adversary $\mathcal{A}$,*

$$\Pr\left[\mathsf{LR}-\mathsf{PKE}-\mathsf{GAME}_{\mathcal{A}}(1^\lambda)=1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

B On [HV25]

The initial version (v1) of our paper on arXiV contained a small gap in last hybrid of our quantum key-fire proof, which was communicated to us by the authors of [HV25] and at the same time they put their paper online. Within a day of receiving their communication, we fixed the small gap in our proof and let them know. Current version of our paper does not have this gap (and it has additional results, such as *LOCC leakage-resilience security* for key-fire and unbounded-message-length incompressibility for OSS) however, we include a discussion here for historical reference.

In [HV25], the authors claimed the following (about the first version of our paper).

1. They claimed that the oracles $\mathcal{D}, \mathcal{D}_0$ that we set to be the signing oracle ($\mathcal{O}_{\mathsf{OSS.Sign}}$) in our incompressible one-shot signature (OSS) scheme construction are not sufficient for signing, and they claimed that thus our claim about existence of 1-bit incompressible OSS was incorrect.

2. They claimed that incompressible OSS for multi-bit messages does not follow directly from incompressible OSS for 1-bit messages.

In the following sections we address both of these points in detail. However, a quick summary is given below:

- The first point mentioned was a misunderstanding by the authors of [HV25] regarding the prior work of [SZ25] and our construction. [HV25] claimed that the oracles $\mathcal{D}, \mathcal{D}_0$ are not sufficient for our signing algorithm, then proceeded to change our signing algorithm (they did not ask us for a clarification) by giving additional access to $\mathcal{P}, \mathcal{P}^{-1}$, and finally, claimed that our argument is flawed since incompressibility cannot hold given access to oracles $\mathcal{P}, \mathcal{P}^{-1}$. However, this is incorrect: It is indeed possible to sign using only the oracles $\mathcal{D}, \mathcal{D}_0$ as shown by various works ([AGKZ20, Shm22, SZ25]).

- The second point was correct and indeed pointed to a small gap in the first version of our key-fire proof: We had implicitly assumed incompressible OSS for multi-bit messages follows in a straightforward manner from incompressible OSS for 1-bit messages, thus we had only given a 1-bit incompressible OSS scheme explicitly.

  However, first, our quantum fire security proof (as opposed to key-fire, which is a slightly stronger primitive) had already relied on only incompressible OSS for 1-bit messages, and thus this gap did not effect our original quantum fire security proof. Second, for proving key-fire, we had relied on incompressible OSS for multi-bit messages only in the last hybrid of the proof. Thus, this was a localized gap and can be fixed it with a small change to make it rely on only 1-bit incompressible OSS. Overall, it did not affect the main ideas and techniques of our paper. We communicated this small fix to them within a day of receiving their email.

More details now follow.

B.1 Signing Oracle

In our incompressible OSS scheme, we have four oracles $\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}, \mathcal{D}_0$. We set $\mathcal{D}$ and $\mathcal{D}_0$ as the signing and signature verification oracles. We proved that in the presence of the signing and verification oracle (that is $\mathcal{D}, \mathcal{D}_0$), our scheme satisfies an incompressibility property.

The authors of [HV25] claimed that $\mathcal{D}, \mathcal{D}_0$ are not sufficient for signing, and they claimed that we must include $\mathcal{P}, \mathcal{P}^{-1}$ to the signing oracle $\mathcal{O}_{\mathsf{Sign}}$. Then they point out that given $\mathcal{P}$, $\mathcal{P}^{-1}$, the incompressibility property is broken. However, indeed the oracles $\mathcal{D}, \mathcal{D}_0$ are sufficient for signing

and we do not need to include $\mathcal{P}, \mathcal{P}^{-1}$. In this version, we added a part that recalls the details of the signing algorithm (which is entirely from the previous work) to prevent any future confusion.

To explain in more detail, in the one-shot signature literature, there are two main/well-known signing algorithms, the so-called Grover-style signing algorithm (from [AGKZ20]), and the generic signing algorithm from [DS23]. The confusion from [HV25] might be due to the fact that we said that our signing algorithm is as in [SZ25], however [SZ25] does not explicitly write out its signing algorithm but rather it is implicit that the signing algorithm of [SZ25] is the Grover-style signing algorithm. In the one-shot signature literature, it is well-known that the Grover style signing algorithm is used when the quantum signature token takes the specific form of a coset state (i.e., a uniform superposition over vectors from some coset), which is the case for [SZ25]. This algorithm (with small modifications) has been used in many OSS works such as [AGKZ20] and [Shm22], and it is well understood that this is the signing algorithm for coset based one-shot signature schemes like [SZ25]. Indeed, another follow-up work to [SZ25] also mentions this Grover-style signing algorithm as the signing algorithm of the [SZ25] construction: See Section 2.3, Page 11 in [BGMS25] where it is also clear that the only oracle being used for signing is $\mathcal{D}$. Note that [BGMS25] was put on arXiv on October 10th, which is prior to the publication of [HV25].

Perhaps another point of the confusion from [HV25] is because [SZ25] essentially constructs both a one-shot signature scheme and a stronger primitive called a non-collapsing collision-resistant hash (non-collapsing CRH). Specifically, [SZ25] cites a known result by [DS23], which says that a non-collapsing CRH generically implies an OSS. However, the context is to claim a stronger primitive (non-collapsing CRH over OSS), and they do not not mean that the specific construction at hand needs to use this generic transformation to get an OSS. Indeed, in the specific case of the construction from [SZ25], their construction is both an OSS directly and a non-collapsing CRH, and the well-known Grover-style signing is to be used for signing when the goal is to obtain OSS directly.

B.2 Multi-bit Incompressible OSS

We showed the existence of incompressible OSS for 1-bit messages. 1-bit OSS is sufficient for most of our proof, but in the first version of our paper, in the proof of our last hybrid in key-fire security, we had invoked incompressible OSS for multi-bit messages. We had implicitly assumed incompressible OSS for multi-bit messages follow in a straightforward manner from incompressible OSS for 1-bit messages, thus we had only given a 1-bit incompressible OSS scheme explicitly. However in a nice observation, [HV25] point out that parallel repetition of a 1-bit OSS scheme to obtain multi-bit OSS scheme, does not preserve incompressibility.

This was a localized issue, and did not affect the rest of our paper. Moreover, this affected only key-fire: Without any changes, our quantum fire security (as opposed to key-fire, which is a slightly stronger primitive) in the first version of our paper still holds using only 1-bit OSS. For key-fire security, by making a small change to our construction and to the last hybrid, we made the proof work using only 1-bit incompressible OSS. When the gap was first pointed out to us by the authors of [HV25], we communicated this fix to them within a day of receiving their email.