Unclonable Cryptography in Linear Quantum Memory*

Omri Shmueli[1] and Mark Zhandry[2]

[1]NTT Research†
[2]Stanford University‡

Abstract

Quantum cryptography is a rapidly-developing area which leverages quantum information to accomplish classically-impossible tasks. In many of these protocols, quantum states are used as long-term cryptographic keys. Typically, this is to ensure the keys cannot be copied by an adversary, owing to the quantum no-cloning theorem. Unfortunately, due to quantum state's tendency to decohere, persistent quantum memory will likely be one of the most challenging resources for quantum computers. As such, it will be important to minimize persistent memory in quantum protocols.

In this work, we consider the case of one-shot signatures (OSS), and more general quantum signing tokens. These are important unclonable primitives, where quantum signing keys allow for signing a single message but not two. Naturally, these quantum signing keys would require storage in long-term quantum memory. Very recently, the first OSS was constructed in a classical oracle model and also in the standard model, but we observe that the quantum memory required for these protocols is quite large. In this work, we significantly decrease the quantum secret key size, in some cases achieving asymptotically optimal size. To do so, we develop novel techniques for proving the security of cryptosystems using coset states, which are one of the main tools used in unclonable cryptography.

*This work is subsumed by [HSVZ26].

†This work was done in part while the author was a research fellow at the Simons Institute for the Theory of Computing at UC Berkeley.

‡Work performed while also at NTT Research.

Contents

1 Introduction **3**
- 1.1 Results . . . 4
- 1.2 Overview of Techniques . . . 5
- 1.3 Concurrent and Independent Work . . . 14

2 Cryptographic Tools **15**
- 2.1 Learning With Errors . . . 17
- 2.2 Two iO Techniques . . . 18
- 2.3 Subspace Hiding Functions . . . 19
- 2.4 Hardness of Concentration in Hidden Random Subspace . . . 23

3 Short One-Shot Signatures Relative to a Classical Oracle **27**
- 3.1 Bloating the Dual . . . 33
- 3.2 Simulating the Dual . . . 36
- 3.3 Hardness of the Dual-free Case from Claw-free Permutations . . . 39

4 Short One-Shot Signatures in the Standard Model **44**
- 4.1 Bloating the Dual . . . 45
- 4.2 Simulating the Dual . . . 51
- 4.3 Hardness of the Dual-free Case from Decomposable Claw-free Functions . . . 55

1 Introduction

Quantum computers will completely upend cryptography. In the short term, quantum algorithms will render insecure all of the public key cryptosystems deployed today. In the longer term, however, quantum computers will enable never-before-possible cryptographic applications. The canonical such application is *unclonable* quantum software [Aar09, ALL+21, LLQZ22], which seeks to make programs uncopyable through no-cloning theorem, a uniquely quantum phenomenon which dictates that it is impossible to copy quantum states. There are numerous works considering special cases or variants of unclonable software in cryptography, where secret keys for signatures, pseudorandom functions, decryption, etc, are made unclonable (e.g. [BDS23, AGKZ20, GZ20, Shm22b, SZ25, CLLZ21, ÇG24]). A strengthening of unclonable software is quantum one-time programs [BGS13, GLR+25], which are not only unclonable, but self-destruct after evaluation. Unclonable software also arises implicitly in the study of obfuscating quantum programs [BKNY23, BBV24, HT25, BGMS25].

Long-term quantum memory. In all of the applications listed above, a key requirement is quantum memory to store the software / secret key. Importantly, this quantum memory needs to be persistent and long term, since in typical applications one aquires the software/key long before it is used. Long-term quantum memory, however, seems very difficult to achieve, as quantum states have a tendency to quickly decohere. Contrast this with short-term, or "online" quantum memory used during quantum computations, which just needs to last as long as the computation. We expect long-term quantum memory to eventually be possible, but we also expect it will be more expensive and more difficult to maintain than either classical or short-term quantum memory.

Therefore, an important metric for protocols leveraging quantum information, and in particular, unclonable cryptosystems, is how much long-term quantum memory is needed.

The quadratic memory barrier. Unfortunately, many works, and in particular all the applications cited above, suffer from very large programs/keys. Namely, the quantum part of the program/key in these works is always at least quadratic in the security parameter λ. This is a result of these works all operating in a bit-by-bit fashion over the program's inputs. Each bit of input is assigned an unclonable quantum state that enables evaluating the program. Here, we assume that the program inputs are at least λ bits. Moreover, in order to maintain unclonability, each of the unclonable quantum states is at least λ bits. The resulting combined quantum states are therefore $\Omega(\lambda^2)$ bits, and other issues could make them even longer[1].

We therefore propose breaking the quadratic long-term quantum memory barrier as an important and interesting goal.

The case of signature tokens and one-shot signatures. In this work, we focus on the class of one-time programs and specifically on the case of quantum signature tokens, first defined by Ben-David and Sattath [BDS23]. A quantum signature token is a secret signing key that allows for signing any message, but after signing the message, the token self-destructs. Formally, the quantum signing key $|\mathsf{sk}\rangle$ is sampled together with a classical public verification key pk, and it is

[1] We note that the above actually suggests quantum states of size $\Omega(n\lambda)$ where n is the input length to the program. However, it is often possible to shrink the quantum state to $\Omega(\lambda^2)$ by only assigning quantum states to a λ-bit hash of the input.

computationally impossible to find two classical signatures that are both verified with respect to the same classical key pk. Such signature tokens are clearly impossible classically, but for signing tokens consisting of quantum states, the act of signing constitutes an irreversible measurement on the token, which may make it incapable of signing future messages. Here, the quantum signing key is generated by a trusted third party.

Amos, Georgiou, Kiayias, and Zhandry [AGKZ20] propose an even stronger notion called one-shot signatures (OSS). Here, the original quantum secret key may be generated by the adversary themselves, and even then, it cannot sign more than a single message. One-shot signatures in particular have numerous applications, such as blockchain-free smart contracts [AGKZ20, Sat22], overcoming lower-bounds in consensus protocols [Dra23], and providing a solution to the blockchain scalability problem [CS20].

In [BDS23], signature tokens are constructed in a classical oracle model. In [AGKZ20], OSS is constructed in an oracle model, but without proof. Very recently, Shmueli and Zhandry [SZ25] construct OSS that is provably secure in a classical oracle model, or secure in the standard model under (somewhat) standard cryptographic assumptions.

Quantum signing keys inherently require $\Omega(\lambda)$ qubits, since an adversary can always brute-force search for the signing key. In all of these constructions, however, the base construction signs only a single bit. We can then upgrade this scheme to sign general messages, by signing each bit separately using a separate key. The result, is quantum signing keys of size $\Omega(\lambda^2)$. In the case of the OSS construction in [SZ25], the signatures are even larger due to additional considerations in their construction/security proof: a token for signing a single bit takes $\Omega(\lambda^2)$ in a classical oracle model, and even larger in the standard model. This boils down to signing keys of overall size $\Omega(\lambda^3)$ for OSS to sign λ-bit messages in a classical oracle model.

1.1 Results

In this work, we overcome the quadratic barrier for long-term quantum memory in both signature tokens and one-shot signatures. First, in a classical oracle model we give constructions with asymptotically-optimal quantum storage $O(\lambda)$ and unconditional security:

Theorem 1. *Relative to a classical oracle there exists secure OSS with $O(\lambda)$-sized quantum secret keys, where each quantum secret key can sign on messages of size λ and for every T and a quantum unbounded algorithm making T queries to the oracles, the probability to find two different signatures corresponding to the same public verification key pk is $\frac{\mathsf{poly}(T)}{2^\lambda}$, for some polynomial $\mathsf{poly}(\cdot)$.*

In the standard model, depending on the computational assumptions used, we also obtain either $O(\lambda)$ or $O(\lambda^2)$-sized quantum secret keys (Theorems 2, 3 and 4).

Our construction is an optimization of the OSS construction of [SZ25], which in turn has a specific structure that allows the sampling of signature tokens in the form of *coset states*. Coset states, in turn, play a major role in quantum cryptography. For example, in all of the above-listed applications of unclonable programs, coset states are the quantum states which sit in long-term quantum memory (and take $\Omega(\lambda)$ qubits each). As a byproduct, we believe these techniques may be useful for quantum memory optimization in other applications built on similar structures, such as quantum copy protection and quantum one-time programs, and leave this as an interesting direction for future work.

Main Technical Contribution. Our results are achieved by two main technical steps, which can be viewed as two separate new techniques. First, we develop a novel quantum algorithm for signing coset states, which signs on an entire message $m \in \{0,1\}^\lambda$ using a single (small) quantum coset state. Our signing algorithm also signs the bits of m in parallel.

The second contribution is a new hardness reduction of the OSS construction of [SZ25], which originally require their generated quantum signature tokens to have size $\Omega\left(\lambda^2\right)$. A central object defined in the proof of [SZ25] are coset partition functions (CPFs), which are the source of the size-$\Omega\left(\lambda^2\right)$ states. Our new reduction introduces a subset of CPFs with additional algebraic structure. Our structured CPFs generally allow for a smaller quantum state ($\approx \lambda$ instead of $\approx \lambda^2$) in reductions, while maintaining the same level of provable security, thus saving a factor of λ in quantum memory and showing that the generated coset states can be of size $O\left(\lambda\right)$.

Combining the above two steps, we are ultimately left with a OSS scheme that generates quantum signing keys of size $O\left(\lambda\right)$ for which it is $2^{\Omega(\lambda)}$-hard to forge a signature. These keys enable signing λ-bit messages, and by a parallel signing algorithm.

1.2 Overview of Techniques

What is λ, anyway? We first make precise what it means to have $O(\lambda)$-bit quantum long-term storage. After all, we could take any scheme with λ^c quantum storage, and simply define $\lambda' = \lambda^c$ as the new security parameter. Since the usual polynomial/negligible way of defining cryptographic notions is robust to polynomial changes in the security parameter, λ' is a valid security parameter, and now the quantum storage is λ', linear in the security parameter. But of course, we didn't actually change the scheme!

Instead, in order to have a robust measure of secret key size in terms of the security parameter, we follow the "bit security" view. Here, λ is set to be such that there are no 2^λ-time attacks. Slightly more formally, λ-bit security means that there is no attack running in time T that succeeds with probability p such that $T/p < 2^\lambda$. Our goal is to achieve $O(\lambda)$-qubit quantum secret keys under this notion of λ.

Put another way, the best attack time (or really, ratio of attack time to success probability) should be exponential in the quantum secret key length.

The OSS of [SZ25]. We briefly recall the version of the OSS scheme of [SZ25] defined relative to an oracle. The oracle comes in three parts:

- An oracle $\mathcal{P}(x)$, which outputs a string y and a vector $\mathbf{u}$. There are two guarantees. First, the map $x \mapsto y$, which we will denote by H, is many-to-1. Second, when restricted to the set of x's such that $H(x) = y$, the map $x \mapsto \mathbf{u}_x$ perfectly embeds the preimage set into an affine subspace S_y. $\mathcal{P}$ is in particular injective.
- An oracle $\mathcal{P}^{-1}\left(y, \mathbf{u}\right)$, which inverts $\mathcal{P}$. $\mathcal{P}^{-1}$ checks that $\mathbf{u} \in S_y$ and inverts only in that case, and outputs $\perp$ otherwise.
- An oracle $\mathcal{D}\left(y, \mathbf{v}\right)$, which checks if $\mathbf{v} \in S_y^\perp$, the subspace dual to S_y.

In [SZ25] it is shown that given the above oracles, the function H is both collision-resistant and non-collapsing. The notion of non-collapsing [Unr16] here, roughly means that given the oracles $\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}\right)$, uniform superpositions over preimage sets of H can be distinguished from classical

states. A non-collapsing CRH generically implies OSS (while the converse is not known), and generally, to obtain OSS from a non-collapsing CRH, one can consider the superposition of preimages as the quantum signing key. However, the specific construction above directly gives an OSS as well, as it samples coset states, which land themselves naturally to known techniques for quantum signing.

To set up a quantum secret key/classical public key pair, one creates the uniform superposition $\sum_{x\in\{0,1\}^n}|x\rangle$ over all possible inputs x, evaluates $H(x)$, and measures to get y. The state then collapses to the uniform superposition over pre-images of y. Then, one can use $\mathcal{P}$ once again to compute a superposition of vectors in the affine space $\sum_{x:H(x)=y}|x,\mathbf{u}_x\rangle$ (or do this already in the first call to $\mathcal{P}$), followed by an application of the inverse $\mathcal{P}^{-1}$, to un-compute the information in the left register, containing the x's. We are left with the state in the form $\sum_{x:H(x)=y}|\mathbf{u}_x\rangle := \sum_{\mathbf{u}\in S_y}|\mathbf{u}\rangle$ which we will call $|\psi_y\rangle$. The public key is $\mathsf{pk} := y$, and the secret key is $|\mathsf{sk}\rangle := |\psi_y\rangle$.

A signature on a bit b is a string $\mathbf{u}_x$ starting with b such that $\mathbf{u}_x \in S_y$ for $\mathsf{pk} = y$. Observe that by measuring $|\mathsf{sk}\rangle$, the signer can obtain a signature on a random bit b. Here, we describe a useful view of how to sign an arbitrary bit; it is not exactly how the prior works sign[2], but will help illustrate some of our ideas. In order to sign an arbitrary bit, measure the first bit of $\mathbf{u}$. If it matches b, the signer measures the rest of $\mathbf{u}$ to get a signature on b. If it doesn't match b, the signer can actually correct the state back to $|\mathsf{sk}\rangle = |\psi_y\rangle$, and try again. This step requires the ability to project onto $|\psi_y\rangle$, which is accomplished with the oracle $\mathcal{D}$, following now-standard techniques [AC12]. Finally, the inability to sign more than one message follows from the collision resistance of H, which [SZ25] prove through a careful reduction to standard quantum collision-resistance results. In fact, the collision-resistance of H proves that the OSS has strong unforgeability, where it is impossible to even find two different signatures for the same message.

Why the large signing keys. We now explore why [SZ25] has large secret keys. Clearly, $\Omega(\lambda)$-qubit keys are necessary. There are two additional sources for the large secret key, resulting in $\Omega(\lambda^3)$-qubit keys:

- To sign large messages, [SZ25] sign bit-by-bit, with each bit having its own key. For messages of length ℓ, this requires secret keys containing at least $\Omega(\lambda\cdot\ell)$ qubits. Note that we can always hash large messages down to λ bits before signing. Thus, we can take $\ell = \lambda$.

- Another source is more subtle. In the security proof, [SZ25] reduce security to the collision-resistance of something they call a coset partition function (CPF). A CPF is a function where all the pre-image sets are affine subspaces. They in particular need a CPF where the preimage sets have size 2^λ. To get such a pre-image set, they start with random 2-to-1 functions, which are automatically CPFs with pre-image sets of size 2, since any two points form a line in $\mathbb{Z}_2^n$. Standard query-complexity lower-bounds show that such 2-to-1 functions are collision resistant, provided the input length is λ bits. In order to get a CPF with 2^λ-sized preimage sets, [SZ25] simply apply λ separate 2-to-1 functions in parallel. This results in a CPF which is collision-resistant but has input of size λ^2. In the security proof, the CPF is embedded inside the oracles $\left(\mathcal{P},\mathcal{P}^{-1},\mathcal{D}\right)$, in a way such that its input size lower-bounds the size of the signing keys. Then, combined with needing $\ell = \lambda$ separate secret keys, the overall key size is $\Omega\left(\lambda^3\right)$.

[2]The only work using this signing technique is [Shm22b].

We now briefly describe how we overcome both of the issues above, to obtain secret keys of size $O(\lambda)$ in the oracle model.

Signing multiple bits using a single quantum state. First, we show how to modify the construction to sign λ bits with one secret key, as opposed to needing λ separate keys. The starting observation is that there is an easy way to extend verification to handle many bit messages: a signature on $m \in \{0,1\}^\lambda$ is just a string $\mathbf{u}$ whose first λ bits are m and such that $\mathbf{u} \in S_y$ for $\mathsf{pk} = y$. The inability to sign multiple messages still follows from the collision resistance of H.

The problem is that the measure-then-correct signing algorithm does not work. The natural generalization of the algorithm is to measure the first λ bits of $\mathbf{u}$, and hope that they match the message; if not, then correct the state back to $|\mathsf{sk}\rangle = |\psi_y\rangle$, and try again. The problem is that there is no granularity here: either the first λ bits of $\mathbf{u}$ match m, or if *any* bit of $\mathbf{u}$ fails to match m, we have to start all over again. The probability that the first λ bits of the measured $\mathbf{u}$ match m is $2^{-\lambda}$, meaning in expectation 2^λ trials are necessary to sign a message[3]. However, we show that by adding extra information to our dual oracle, we are able to sign in polynomial time.

To illustrate, for $i \in [\lambda]$ and an i-bit string x, let $|\psi_{y,x}\rangle$ be the uniform superposition over S_y whose initial i bits are exactly x. Then $|\psi_{y,m}\rangle$ is the state we want to ultimately obtain, as a measurement on it will give us a signature for m. The state that we start with, which equals to our quantum signing key, is $|\psi_{y,\emptyset}\rangle$. Then by the same measure-then-correct approach, we can obtain the state $|\psi_{y,m_1}\rangle$, where m_1 is the first bit of m. Now, we want to perform a fresh round of measure-then-correct, but where we start from $|\psi_{y,m_1}\rangle$ and move to $|\psi_{y,m_1m_2}\rangle$. The probability of success in the measuring step is $1/2$, so the expected number of trials is 2, which is efficient. We can continue in this way, gradually moving to $|\psi_{y,m_1m_2m_3}\rangle, \cdots$ all the way to $|\psi_{y,m}\rangle$, and each step is efficient since in each step, we are only measuring one bit, and so in expectation only need two trials per step.

The issue is that, in order to correct to the states $|\psi_{y,m_1}\rangle, |\psi_{y,m_1m_2}\rangle, \cdots$, we need a mechanism to project onto those states efficiently. Unfortunately, the existing oracles do not provide this. However, we can modify the oracles to accomplish this. Namely, to project onto $|\psi_y\rangle$, we needed two oracles, one for S_y and one for $S_y^\perp$. In order to project onto $|\psi_{y,m_1}\rangle$, we also need two oracles. One is for S_{y,m_1}, the affine subspace of S_y whose first bit is m_1. The other is $S_{y,m_1}^\perp$, the dual to S_{y,m_1}.

Fortunately, we actually already have S_{y,m_1} for free: membership in S_{y,m_1} is checked by checking membership in S_y and then checking that the first bit equals m_1. However, $S_{y,m_1}^\perp$ is a super-space of $S_y^\perp$, and we do not (yet) have access to this oracle. Our solution is to provide it. Now, as we go to more and more bits of m, there will be more and more subspaces $S_{y,m}^\perp$ we would have to provide oracles for. Eventually this will be exponentially many subspaces. This may actually be fine to provide access to all these subspaces, but we do not know how to prove it. Instead, we observe that the different choices of subspace $S_{y,m}^\perp$ as m varies actually have predictable relationships to each other. It actually suffices to give out membership checks for $S_{y,0}^\perp, S_{y,00}^\perp, \cdots, S_{y,0^\lambda}^\perp$, and this will allow for checking membership in all the $S_{y,m}^\perp$ subspaces. Note that adding these oracles allows for efficient signing, but we can no longer use the security proof from [SZ25] as a black box due to the extra oracles. In fact, our actual signing algorithm, which we describe next, even needs more information, so we will later touch upon how we modify this.

[3] An additional issue is that the time to correct also grows exponentially with λ.

Parallel Signing. One drawback of the above approach is that it makes the signing process sequential, in that the signing of bit i of m has to be processed before the next bit $i+1$ can be processed. This is because, for the projection on the $(i+1)$-th state $|\psi_{y,m_1\cdots m_{i+1}}\rangle$ to have constant success probability, we need to already be in the state $|\psi_{y,m_1\cdots m_i}\rangle$. Contrast with the original signing procedure which just signs each bit in parallel (but using many quantum states).

We overcome this by actually modifying the signing process above to make it highly parallelizable. We measure *all* the first λ bits of $\mathbf{u}$. With overwhelming probability, roughly half of them will match the corresponding bit of m. So we keep those bits, but correct all the bits that did not match, and do this correction in parallel. This requires significant care, and there are two main problems to overcome. First, the iterative signing process outlined above, needed to give out membership checks for the subspaces $S^{\perp}_{y,0}, S^{\perp}_{y,00}, S^{\perp}_{y,000}, \cdots$. But these subspaces, by depending on ever-larger prefixes, *only* allow for signing the bits in sequential order. When we measure the first λ bits of $\mathbf{u}$, the positions that do not match m will be in random positions, and the existing structure does not allow for directly correcting those bits. Instead, we give out a small collection of large subspaces, whose *intersections* allow us to simulate the exponentially-many subspaces, which are needed in order to fix any possible bit positions. Specifically, we give access to λ distinct subspaces of $S^{\perp}_{y,0^\lambda}$, all of which have dimension one smaller than $S^{\perp}_{y,0^\lambda}$. The differing intersections of these, turn out to be sufficient to correct any error.

So, we have the information we need for correcting, and now turn to computational complexity, which is where the second difficulty. Naively correcting the $\approx \lambda/2$ bits that didn't match the message-to-sign, and transforming them back into the uniform superposition seems to require exponential time, since the time to correct grows exponentially with the number of bits that need correcting. However, we show that we can efficiently correct them to uniform superposition with a random choice of phase; the phase turns out not to matter, since the next step is to just measure the bits anyway. What is important is that measuring after correcting re-randomizes these bits. Therefore we measure the bits again, and at this point, roughly $3\lambda/4$ of the bits will match m. Another round gives $7\lambda/8$, and so on.

We can continue in this way, until we get all the bits of m to match. This would require roughly $\log \lambda$ sequential rounds of queries. But we can do better: instead of signing m directly, we instead map m to a codeword c of a good error correcting code, and sign c. We then relax the condition on $\mathbf{u}$ being a valid signature to only require the first λ bits of $\mathbf{u}$ to have a Hamming distance at most, say, $\lambda/6 > \lambda/8$ away from the codeword c. With overwhelming probability, our signing process above, when run for sequential 3 steps, will produce such a valid signature.

As for security, assuming the code has minimum distance strictly greater than $\lambda/3$, for any $\mathbf{u}$, there will be at most one codeword c within the required distance of $\mathbf{u}$. Thus, valid signatures on two distinct messages will lead to two distinct $\mathbf{u}$ that collide, contradicting the collision resistance of H. Note that a random linear code will obtain such a minimum distance with high probability, as long as we set the rate to be an appropriately small constant.

Finally, as mentioned, the information our oracles provide is significantly more than just giving out membership access to S_y, $S^{\perp}_y$, and the previous security analysis of [SZ25] cannot be used as is. By a new analysis we show that even with this extra information, it is impossible to find a collision in H. In the process, we also define and prove the security of a generalization of subspace-hiding obfuscation, which we refer to as subspace-hiding functions in this work (see Section 2.3).

Overcoming the CPF input-size Problem. We go back to thinking about the size of the quantum signing key for signing λ-bit messages. Until now, we explained how to go from keys of size $\Omega\left(\lambda^3\right)$ qubits, due to the original construction of [SZ25], to $O\left(\lambda^2\right)$ in our optimized construction. Next, we turn to overcoming the issue with the coset partition function (CPF) input size. Looking at the final step of the security reduction of [SZ25], what is shown is that given only the oracles $\mathcal{P}$, $\mathcal{P}^{-1}$ (that is, without the dual oracle $\mathcal{D}$), the function H is collision resistant. The central tool of the reduction is a CPF $\mathcal{Q}$ from n to r bits, where the size of each preimage coset of $\mathcal{Q}$ is 2^λ and $\mathcal{Q}$ is collision-resistant. Very roughly, the reduction embeds $\mathcal{Q}$ inside both oracles $\mathcal{P}, \mathcal{P}^{-1}$, in an indistinguishable way, to later claim that a collision in H constitutes a collision in $\mathcal{Q}$. In order to claim indistinguishability between the original oracles $\mathcal{P}, \mathcal{P}^{-1}$ and the oracles where $\mathcal{Q}$ is embedded in, there are two primary arguments where the structure of $\mathcal{Q}$ comes up.

First, when computing y i.e., the first part of the output of $\mathcal{P}$, the original $\mathcal{P}$ executes a random permutation Π on the input $x \in \{0,1\}^n$ and $H(x) := y \in \{0,1\}^r$ is given by the first r output bits of the permutation's output. In order to embed $\mathcal{Q}$ indistinguishably, [SZ25] observe that a CPF from n to r bits can always be extended to a permutation in the following way: the first r bits are simply the output of $\mathcal{Q}$ which tells us what coset the input element belonged to, and the remaining $n-r$ bits can serve as the coordinates vector $\mathbf{z} \in \mathbb{Z}_2^{n-r}$ of that element with respect to the coset of the output y – recall that because $\mathcal{Q}$ is a CPF, the preimage set of each y is an affine subspace, and thus every element has a coordinates vector. Finally, composing a random permutation with any fixed permutation is statistically equivalent to just a random permutation, so, the permutation that we extended $\mathcal{Q}$ to, can be composed with the random permutation Π inside the original $\mathcal{P}$. This constitutes that for the first part of the output of $\mathcal{P}$ (and in fact, also for the inverse $\mathcal{P}^{-1}$), $\mathcal{Q}$ can be put inside the oracles in an indistinguishable manner.

The second place where the structure of $\mathcal{Q}$ is used is to simulate the vectorial part of the output of $\mathcal{P}$, that is, the vectors $\mathbf{u}_x \in S_y$. Here, the original $\mathcal{P}$ samples a random affine subspace S_y given each output y. Now, recall that due to the inverse oracle $\mathcal{P}^{-1}$ there should be an *invertible* mapping between the inputs $x \in \{0,1\}^n$ and the vectors $\mathbf{u}_x \in S_y$, so, the affine subspaces in the security reduction to the collision-resistance of $\mathcal{Q}$ cannot just be sampled independently of $\mathcal{Q}$. There needs to be an efficient invertible mapping between the simulated affine subspaces and the actual preimage sets of $\mathcal{Q}$. Fortunately, the structure of the CPF can be used again, but in a different manner: in order to simulate a random affine subspace, it turns out it is enough to take the preimage set elements of $\mathcal{Q}$ for some output y, and multiply them by the same random full-rank matrix (that is, a random i.i.d. matrix for every output y of $\mathcal{Q}$). Intuitively, the multiplied element can now be thought of as an element in a random affine subspace, and the reason is that the original preimage set of $\mathcal{Q}$ was an affine subspace already, and multiplying any affine subspace by a random matrix gives a *random* affine subspace. Finally, since the reduction is the one sampling the random matrix, the mapping between affine subspace elements from the simulated S_y, and elements from the preimage set containing the x's, is efficiently computable.

Let us denote by n the input size to $\mathcal{P}$ and by k the size of the vectors in the affine subspaces S_y. Note that in the above reduction, the input size to $\mathcal{Q}$, which we denote by n', lower bounds both n and k. This means that, if we had a way to construct a CPF with input size $n' \approx \lambda$ such that $\mathcal{Q}$ is 2^λ-collision-resistant[4], we would be done. Unfortunately, we only know how to build a CPF with input size λ^2 that is 2^λ-collision-resistant. As mentioned earlier in the introduction, this

[4]That is, where the probability for any quantum algorithm making a polynomial number of queries to find a collision is $2^{-\Omega(\lambda)}$

is done by taking the parallel repetition of a random 2-to-1 function. It seems plausible that such CPFs may exist, and in fact [AGKZ20] provide a CPF which could reasonably be conjectured to have the desired properties. However, we do not know how to prove such a result.

Folding Coset Partition Functions. We show a new technique to construct a CPF such that even if its input is large, for every input there is a *folded version* of that input. Furthermore there is an invertible mapping between folded and original inputs, which does not break the collision-resistance of $\mathcal{Q}$. Concretely, our folding CPF is given by two algorithms

$$\mathcal{Q} : \mathbb{Z}_2^{\lambda^2} \to \left(\mathbb{Z}_2^{\lambda^2-\lambda} \times \mathbb{Z}_2^{2\cdot\lambda-1}\right), \quad \mathcal{Q}^{-1} : \left(\mathbb{Z}_2^{\lambda^2-\lambda} \times \mathbb{Z}_2^{2\cdot\lambda-1}\right) \to \mathbb{Z}_2^{\lambda^2} \ .$$

In the previous work, the CPF $\mathcal{Q} : \mathbb{Z}_2^{\lambda^2} \to \mathbb{Z}_2^{\lambda^2-\lambda}$ was given by taking a parallel repetition of λ functions, each is a random 2-to-1 function from λ to $\lambda - 1$ bits. Here, the first step to our solution will be not to take arbitrary 2-to-1 functions but *claw-free permutations*. A claw-free permutation $L : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda-1}$ is given by a pair of random permutations Π_0, Π_1 such that $L\left(b \in \{0,1\}, x \in \{0,1\}^{\lambda-1}\right) := \Pi_b(x)$. Claw-free permutations can easily be proved as collision-resistant in a classical oracle model (and we give a proof in the body of our paper). A nice property of these functions is that collisions always differ in the first bit. In particular, taking the view of claw-free permutations as 1-dimensional CPFs, one can think about the first bit as containing the information of the coordinates vector of that coset element in the preimage set. This coordinate information is computable publicly for a claw-free permutation, and this will be useful for our new reduction.

Our function $\mathcal{Q} : \mathbb{Z}_2^{\lambda^2} \to \left(\mathbb{Z}_2^{\lambda^2-\lambda} \times \mathbb{Z}_2^{2\cdot\lambda-1}\right)$ will be computed as follows: The first part of the output will be a parallel repetition of a claw-free permutation, and accordingly takes $\lambda \cdot (\lambda - 1) = \lambda^2 - \lambda$ bits. The second part of the output will itself contain two parts: the first part is λ bits, and contains the first input bit of each of the λ inputs to the claw-free permutations. The second part (of the second part) is the *sum* of the rest of the inputs. That is, if $\mathbf{w} = (b_1, \mathbf{w}_1, \cdots, b_\lambda, \mathbf{w}_\lambda)$ is the input for $\mathcal{Q}$ broken down to the λ inputs to the claw-free permutations, then we are thinking of $\overline{\mathbf{w}} := \sum_{j\in[\lambda]} \mathbf{w}_j$. The overall output of $\mathcal{Q}$ is

$$(y_1, \cdots, y_\lambda) \in \{0,1\}^{\lambda^2-\lambda}, (b_1, \cdots, b_\lambda, \overline{\mathbf{w}}) \in \{0,1\}^{2\cdot\lambda-1} \ .$$

The inverse function $\mathcal{Q}^{-1}$ can be computed as follows. We have the permutation pairs

$$\left((\Pi_{1,0}, \Pi_{1,1}), \cdots, (\Pi_{\lambda,0}, \Pi_{\lambda,1})\right)$$

defining our function $\mathcal{Q}$, and we can think about the inverse permutations of these pairs

$$\left(\left(\Pi_{1,0}^{-1}, \Pi_{1,1}^{-1}\right), \cdots, \left(\Pi_{\lambda,0}^{-1}, \Pi_{\lambda,1}^{-1}\right)\right)$$

as the "secret key". We use this secret key in order to implement $\mathcal{Q}^{-1}$ in a straightforward way: For each $j \in [\lambda]$, we have the j-th output $y_j \in \{0,1\}^{\lambda-1}$ and the bit b_j, so to invert we compute $b_j, \Pi_{j,b_j}^{-1}(y_j)$. For this, we do not even use the sum part $\overline{\mathbf{w}}$ of the input to $\mathcal{Q}^{-1}$.

The key insight to our reduction, is that the oracles $\mathcal{Q}$, $\mathcal{Q}^{-1}$ can both be perfectly simulated *even if one of the instances of L does not contain the secret key.* In a nutshell, assume that you want to simulate $\mathcal{Q}$, $\mathcal{Q}^{-1}$, and for some instance $i^* \in [\lambda]$ of the claw pairs, you do have the forward

computation $\Pi_{i^*,0}, \Pi_{i^*,1}$ but not the inverse pair $\Pi^{-1}_{i^*,0}, \Pi^{-1}_{i^*,1}$. We will want to find a collision in that instance. Computing $\mathcal{Q}$ in this setting is still easy: we just compute all claw pairs in the forward direction and then for the second part of the output we output the first bits of the corresponding inputs, and also sum the rest of the parts, as described above. The simulation of $\mathcal{Q}^{-1}$ shows where the sum of the inputs becomes handy: for all instances $j \in [\lambda] \setminus \{i^*\}$ we compute the inverses as before – thanks to the claw-free permutations only needing the output y_j and the bit telling us which of the two inputs was it, we recover the corresponding $\{\mathbf{w}_j\}_{j\in[\lambda]\setminus\{i^*\}}$. Finally, all we need to do to get the last $\mathbf{w}_{i^*}$ is by subtracting all of the $\lambda - 1$ elements we obtained, from the sum we got as input to $\mathcal{Q}^{-1}$:

$$\mathbf{w}_{i^*} \leftarrow \overline{\mathbf{w}} - \sum_{j\in[\lambda]\setminus\{i^*\}} \mathbf{w}_j \ .$$

The last observation to make is that the folded part actually structures a coset as well. In the body of the paper we further explain how the folded inputs do not only constitute an invertible map (while keeping $\mathcal{Q}$ collision-free), but these sets also form affine spaces, just smaller ones. In particular, note that the second output of $\mathcal{Q}$ takes only $2\lambda - 1 \in O(\lambda)$ bits. Circling back to the reduction of the collision-resistance of $\mathcal{P}, \mathcal{P}^{-1}$, we still take the λ^2-bit outputs of $\mathcal{P}$ as the input of $\mathcal{Q}$, but now can use the folded part, taking only $O(\lambda)$ bits as vectors to simulate the affine spaces S_y. Since we have the invertible mapping $\mathcal{Q}$, $\mathcal{Q}^{-1}$ we also can invert between $\mathcal{P}$ and $\mathcal{P}^{-1}$, but we shaved a factor of λ from the vectorial part. Since our signing keys are ultimately the vectorial part, we get signature tokens of size $O(\lambda)$.

Standard Model Construction. We now turn to constructing OSS in the standard model, with small quantum secret keys. It is straightforward to adapt all of our techniques to the standard-model construction of [SZ25] to get a secure construction. However, it turns out it is not immediate that the construction has linear-sized quantum secret keys.

To see the issue, recall that we are setting λ to be such that all T time attacks with success probability p have $T/p \geq 2^\lambda$. In particular, if secret keys are length $O(\lambda)$, this means all attacks must take exponential time and/or have exponentially-small success probability, as functions of the length of the secret key.

In the oracle version above, we were able to achieve this due to exponential bounds on the success probability of bounded-query algorithms. In the standard model, achieving such a result requires in particular the hardness of each of the cryptographic building blocks against attacks running in time 2^λ.

Note that the various building blocks may be instantiated with different security parameters. For example, we may set the security parameter for indistinguishability obfuscation to $\lambda_{\mathsf{iO}} = \lambda^c$ for some large constant c [5]. In this case, we only need that the iO is secure against attacks running in time $2^\lambda = 2^{\lambda_{\mathsf{iO}}^{1/c}}$; that is, we only need subexponential-time security of the iO. However, setting the security parameter for components in this way will affect the sizes of certain parameters. For iO, fortunately setting the security parameter in this way only affects the size of the obfuscated program that is a part of the common reference string, and not the quantum secret key size.

The takeaway, however, is that (1) all the primitives need to be at a bare minimum subexponentially-secure[6], and (2) we need to be extremely careful about how the various components interact with the parameter sizes.

[5] In fact, this already happens in the existing proof of [SZ25].

[6] If a primitive P is only $2^{\lambda_P^{o(1)}}$-secure, then in order to achieve $2^{\lambda_P^{o(1)}} \geq 2^\lambda$, we must set λ_P to be super-polynomial

Recall the assumptions used to instantiate [SZ25]:

- Indistinguishability obfuscation (iO). They already need sub-exponential security for their proof, and as discussed above, sub-exponential security suffices for us.

- A sub-exponentially-secure one-way function. The one-way function arises primarily in two places. One is for constructing an object called a permutable PRP, an analog of punctured PRFs for permutations. We note that for this application sub-exponential security still suffices for us, as it will blow up the key but not the block size of the PRP; the key blows up the size of the obfuscated program, which only affects the CRS.

  The other application is to implement what they call a sparse trigger. Here, there is a bad event (elaborated on below) which occurs on an α-fraction of inputs. The bad inputs are hidden by mapping this to a $\log(1/\alpha)$-bit string, which in turn serves as input to a pseudorandom generator. Remember that the PRG needs to be secure against 2^λ-time attacks. If the PRG is exponentially secure, we can handle $\alpha = 2^{-O(\lambda)}$, but if the PRG is only sub-exponentially secure, this means α must be correspondingly smaller. This will come up when we discuss the next assumption.

- The polynomial hardness of LWE with sub-exponential noise/modulus ratio. Here, we certainly need to upgrade this to subexponential hardness[7]. But it actually gets worse. LWE is used in two places. First, it is used to implement a lossy function in order to "bloat" the subspaces $\mathcal{D}$ accepts. In this step, we incur a loss in the security reduction equal to the range of the lossy function in lossy mode. On the other hand, we are reducing to the hardness of a one-way function whose effective security parameter is limited by the dimension of the ambient space. But remember that our secret keys are now superpositions of vectors that live in this space. As such, for linear-sized secret keys, we need the lossy function to have range size $2^{O(\lambda)}$ while still having security against algorithms running in time $2^{O(\lambda)}$. Unfortunately, LWE is not secure in this regime, due to attacks running in time $2^{O(\lambda/\log\lambda)}$ where λ represents the bit-length of the secret key[8]. Fortunately, by slightly blowing up the key size to be quasi-linear $\tilde{O}(\lambda)$, we can account for such attacks, though we need the very strong assumption of near-exponentially-secure LWE.

  The more problematic part is where LWE is used to realize a collision-resistant approximately 2-to-1 function. This construction has errors (namely, places where it is not 2-to-1) at a rate of roughly the noise-modulus ratio of LWE. Thus, in light of the sparse trigger requirements, we need the noise-modulus ratio to be on the order of 2^λ. It turns out that setting the noise modulus ratio this high blows of the input/output size of the 2-to-1 function without increasing hardness. The result is that the input/output size of the 2-to-1 function is on the order at least $\Omega(\lambda^3)$, even assuming a security level for LWE that exactly matches the best-known attacks. By carefully modifying the proof we can get the "effective" size to be $O(\lambda^2)$, but we do not know how to shave off the last factor of λ, preventing us from obtaining linear-sized quantum secret keys.

in λ. This would cause some system components to have super-polynomial run-time and size.

[7] As in, we need to assume that LWE with a subexponential noise/modulus ration is subexponentially hard.

[8] LWE attacks run in time $2^{O(n)}$ for dimension n. But the secret is a vector in $\mathbb{Z}_q^n$ which has bit-length $n \log q$. Since q is typically polynomial in λ, this gives $n = \Omega(\lambda \log \lambda)$.

It turns out this issue with the 2-to-1 functions is the only barrier to achieving quasi-linear-sized quantum secret keys in the standard model. If we just accept this limitation, we are able to obtain:

Theorem 2. *Assume all of the following: (1) sub-exponentially- secure indistinguishability obfuscation, (2) exponentially-secure one-way functions, and (3) "optimally-secure" LWE. Then, there exists OSS with $O(\lambda^2)$-sized quantum secret keys, that can sign on messages of size λ and such that for every T-time quantum adversary, the probability to break strong unforgeability is bounded by $\frac{T}{2^{\Omega(\lambda)}}$.*

Here, "optimal" LWE means (1) a noise modulus ratio of $2^{O(\sqrt{n \log q})}$ such that the constant hidden by the Big-Oh is sufficiently small to block known attacks, and (2) there are no attacks with T/p smaller than $2^{O(n)}$. We will set $n = O(\lambda)$. Note that for polynomial noise, we have $q \approx 2^{O(\sqrt{n \log q})}$, and then (1) translates to $q = 2^{O(n)} = 2^{O(\lambda)}$, which is needed for the sparse trigger. (2) is needed for simply to obtain security against algorithms running in time $2^{\lambda} = 2^{O(n)}$.

Alternatively, if we just assume the existence of an exponentially-secure perfect 2-to-1 trapdoor function, we avoid the 2-to-1 function issue entirely, and obtain:

Theorem 3. *Assume all of the following: (1) sub-exponentially-secure indistinguishability obfuscation, (2) exponentially-secure one-way functions, (3) $2^{O(\lambda_{LWE}/\log \lambda_{LWE})}$-secure LWE with a polynomial noise-modulus ratio, and (4) perfect 2-to-1 trapdoor functions with exponentially-secure collision-resistance, which are "permutation-indistinguishable". Then, there exists OSS with $\tilde{O}(\lambda)$-sized quantum secret keys, that can sign on messages of size λ and such that for every T-time quantum adversary, the probability to break strong unforgeability is bounded by $\frac{T}{2^{\Omega(\lambda)}}$.*

Note that (4) implies (2). Here, $2^{O(\lambda_{LWE}/\log \lambda_{LWE})}$-secure LWE is a relaxation of optimal LWE to the case of polynomial q, requiring that security scales exponentially in the dimension n, while the secrets have size $\lambda_{LWE} = n \log q = O(n \log n)$. We will set $n = O(\lambda)$, giving $\lambda_{LWE} = O(\lambda \log \lambda)$, so that we achieve security against 2^{λ}-time attacks.

Being "permutation-indistinguishable" is a mild technical condition required to be compatible with the proof of security using permutable PRPs, which is formalized in Definition 35. We do not know any provably secure constructions satisfying (4), but a simple conjectured such function is the following. Take two PRPs P_{k_0}, P_{k_1} with random keys k_0, k_1, and obfuscate the function mapping $(b, x) \mapsto P_{k_b}(x)$. This is a 2-to-1 function. k_0, k_1 is also a trapdoor, since applying $P^{-1}_{k_0}(y)$ and $P^{-1}_{k_1}(y)$ gives the two pre-images of y. We could in particular set P to be a permutable PRP constructed in [SZ25], which is permutation-indistinguishable. Assuming ideal obfuscation, such an obfuscation of P would in fact be exponentially collision-resistant (assuming the underlying one-way functions are sub-exponentially secure). A common heuristic is to assume that indistinguishability obfuscation (iO) actually acts as an ideal obfuscator. While this heuristic is false in general [BGI+01], it seems reasonable to apply it to "natural" programs that were not explicitly designed to demonstrate an impossibility. So it isn't immediately impossible that using iO would also give this level of security[9]. However, it seems unlikely that we could actually prove such a construction is secure just based on iO and one-way functions, since in particular this implies collision-resistance, which is likely not provable from these assumptions alone [AS15]. Note that still, it might be possible to prove the security of this exact construction as is, using more assumptions. For example, assuming collision-resistant primitives which could possibly be hidden,

[9]Note that we certainly need sub-exponential iO, but do not necessarily need exponential iO, since again it only affects the obfuscated program size and not the size of the quantum secret key.

indistinguishably under the iO, and come up only in later hybrids (which is the same technique used to prove the collision-resistance of our one-shot signature scheme, based on LWE).

Finally, we can even get truly linear secret keys if we eliminate the use of LWE all together, and additionally assume an exponentially-secure *lossy* function:

Theorem 4. *Assume all of the following: (1) sub-exponentially-secure indistinguishability obfuscation, (2) exponentially-secure one-way functions, (3) exponentially-secure lossy functions, and (4) perfect 2-to-1 trapdoor functions with exponentially-secure collision-resistance, which are "permutation-indistinguishable". Then, there exists OSS with $O(\lambda)$-sized quantum secret keys, that can sign on messages of size λ and such that for every T-time quantum adversary, the probability to break strong unforgeability is bounded by $\frac{T}{2^{\Omega(\lambda)}}$.*

A lossy function [PW08] is a function that comes in two modes; an injective mode and a lossy mode. In the injective mode the function is injective, while in the lossy mode the function has a small image of size at most $2^{O(\lambda)}$. It is straightforward to construct lossy trapdoor function using ideal obfuscation and sub-exponentially-secure PRFs: the injective mode is just the obfuscation of an injective PRF, and the lossy mode is the composition $F \circ G$ of two PRFs, where the image of G has size $2^{O(\lambda)}$. Therefore, we can plausibly instantiate (3) by applying iO. As before, while the construction itself may have a security proof, it seems unlikely that this could be proved without using more assumptions, given that lossy functions imply collision-resistance.

1.3 Concurrent and Independent Work

In a concurrent and independent work, [HV25] also consider the size of the quantum secret key in [SZ25]. Their work contains a few results, some of which partially overlap with ours.

OSS in the oracle setting. In the oracle setting, they are able to get the key size down to $O(\lambda^2)$. They shave one multiplicative factor λ relative to [SZ25], by signing all the bits of the message using a single token rather than signing individual bits using many tokens. However, their result got stuck at λ^2 because they were not able to overcome the CPF size bound as we do. Their signing algorithm is sequential, unlike our parallel algorithm.

Remark 5. *They actually claim that the oracle-model construction of [SZ25] requires secret keys of size $\Omega(\lambda^4)$, while it requires $\Theta\left(\lambda^3\right)$. The reason for the λ^4 is when one uses the generic transformation from non-collapsing collision-resistant hash functions to OSS [DS23]. However, while [SZ25] cite [DS23] as a motivation in the overview, the actual construction signs messages directly as opposed to going through [DS23]. As such, this factor of λ is not actually relevant.*

OSS in the plain model. Then, moving from the oracle model to the plain model, [HV25] claim that they achieve the same $O\left(\lambda^2\right)$-qubit secret keys in the plain model, by adapting the security proof of [SZ25] to their modifications. The result is a claim of OSS with $O\left(\lambda^2\right)$-qubit secret keys under the same assumptions as [SZ25]:

Claim 6 (Claimed in [HV25])**.** *Assuming (1) subexponentially-secure indistinguishability obfuscation, (2) subexponentially- secure one-way functions, and (3) (polynomially-secure) LWE with a subexponential noise-modulus ratio, there exists a secure OSS scheme with perfect correctness and $O(\lambda^2)$-bit signatures and $O(\lambda^2)$-qubit signing keys for $\mathsf{poly}(\lambda)$-bit messages.*

Unfortunately, Claim 6 is false: while the security proof is valid for justifying the security of the OSS scheme against polynomial-time attacks, it does not actually justify $O(\lambda^2)$-qubit secret keys. The reason is that they only prove that the construction is polynomially-secure (as was done in [SZ25]), without paying attention to the fact that claiming λ-bit security – as required to meaningfully discuss key sizes as a function of λ – requires security against 2^λ-time attackers. In particular, Claim 6 cannot possibly be true, for *any* fixed polynomial-length secret key, for the simple reason that any such claim inherently requires all building blocks to be sub-exponentially secure,[10] but Claim 6 only assumes the polynomial hardness of LWE. Digging deeper, they also miss the fact that using sub-exponential hardness for the sparse trigger blows up the secret key size. Moreover, even upgrading to "optimal" LWE still further blows up the secret key size, as discussed above.

A straightforward adaptation of their proof, using the assumptions as we do – namely, exponentially-secure one-way functions and same "optimal" LWE assumption – results in their quantum secret keys having size $\Omega(\lambda^4)$, as opposed to our $O(\lambda^2)$. Again, one factor of λ improvement in our work comes from the same factor of λ improvement from the oracle setting, and the other factor of λ comes from our trick of only using "effective" input size in the LWE-based 2-to-1 functions.

Perfect correctness. [HV25] show how to make the OSS construction perfectly correct, which we do not consider. Our parallel signing algorithm has an exponentially-small correctness error of $2^{-\Omega(\lambda)}$. An interesting open question is whether our parallel signing algorithm can be adapted to be perfectly correct.

Quantum fire in the oracle model. Lastly, [HV25] also discuss applications to quantum fire – quantum states that can be cloned but not sent over classical communication – which were recently explored in [NZ24, BNZ25, ÇGS25]. We do not consider this setting in the current work.

2 Cryptographic Tools

We use the following known primitives and notions. Both are implicitly classical primitives with security holding against quantum algorithms.

Definition 7 (Puncturable PRFs)**.** *A puncturable pseudorandom function (P-PRF) is a pair of efficient algorithms* $(\mathsf{F}, \mathsf{Punc}, \mathsf{Eval})$ *with associated output-length function* $m(\lambda)$ *such that:*

- $\mathsf{F} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^{m(\lambda)}$ *is a deterministic polynomial-time algorithm.*
- $\mathsf{Punc}(k, S)$ *is a probabilistic polynomial-time algorithm which takes as input a key* $k \in \{0,1\}^\lambda$ *and a set of points* $S \subseteq \{0,1\}^*$*. It outputs a punctured key* k^S*.*

[10] Since, even if you set the security parameter of the underlying component to be some polynomial κ in λ, this still means the attack runs in time 2^{κ^c}. Polynomial security may only guarantee a security against a running time of $2^{\log^2 \kappa}$.

- $\mathsf{Eval}\left(k^S, x\right)$ *is a deterministic polynomial-time algorithm.*
- **Correctness:** *For any* $\lambda \in \mathbb{N}$, $S \subseteq \{0,1\}^*$, $k \in \{0,1\}^\lambda$, $x \notin S$, *and* k^S *in the support of* $\mathsf{Punc}\left(k, S\right)$, *we have that* $\mathsf{Eval}\left(k^S, x\right) = \mathsf{F}\left(k, x\right)$.
- **Security:** *For any quantum polynomial-time algorithm* $\mathcal{A}$, *there exists a negligible function* ϵ *such that the following experiment with* $\mathcal{A}$ *outputs 1 with probability at most* $\frac{1}{2} + \epsilon\left(\lambda\right)$*:*
  - $\mathcal{A}(1^\lambda)$ *produces a set* $S \subseteq \{0,1\}^*$.
  - *The experiment chooses a random* $k \leftarrow \{0,1\}^\lambda$ *and computes* $k^S \leftarrow \mathsf{Punc}\left(k, S\right)$. *For each* $x \in S$, *it also sets* $y_x^0 := \mathsf{F}\left(k, x\right)$ *and samples* $y_x^1 \leftarrow \{0,1\}^{m(\lambda)}$ *uniformly at random. Then it chooses a random bit* b. *It finally gives* $k^S, \{\left(x, y_x^b\right)\}_{x \in S}$ *to* $\mathcal{A}$.
  - $\mathcal{A}$ *outputs a guess* b' *for* b. *The experiment outputs 1 if* $b' = b$.

Different security levels. For arbitrary functions $f_0, f_1 : \mathbb{N} \to \mathbb{N}$, we say that the P-PRF is $\left(f_0, \frac{1}{f_1}\right)$-secure, if in the above security part of the definition, we ask that the indistinguishability holds for every adversary of size $\leq f_0(\lambda)$ and we swap $\epsilon(\lambda)$ with $\frac{1}{f_1(\lambda)}$. Concretely, a sub-exponentially secure P-PRF scheme would be one such that there exists a positive real constant $c > 0$ such that the scheme is $\left(2^{\lambda^c}, \frac{1}{2^{\lambda^c}}\right)$-secure.

Definition 8 (Indistinguishability Obfuscation (iO))**.** *An indistinguishability obfuscator (iO) for Boolean circuits is a probabilistic polynomial-time algorithm* $\mathsf{iO}\left(\cdot,\cdot,\cdot\right)$ *with the following properties:*

- **Correctness:** *For all* $\lambda, s \in \mathbb{N}$, *Boolean circuits* C *of size at most* s, *and all inputs* x *to* C,

$$\Pr\left[\mathsf{O}_C(x) = C(x) \ : \ \mathsf{O}_C \leftarrow \mathsf{iO}\left(1^\lambda, 1^s, C\right)\right] = 1 \ .$$

- **Security:** *For every polynomial* $\mathsf{poly}(\cdot)$ *there exists a negligible function* ϵ *such that the following holds. Let* $\lambda, s \in \mathbb{N}$, *and let* C_0, C_1 *two classical circuits of (1) the same functionality (i.e., for every possible input they have the same output) and (2) both have size* $\leq s$.

$$\left\{\mathsf{O}_{C_0} \ : \ \mathsf{O}_{C_0} \leftarrow \mathsf{iO}\left(1^\lambda, 1^s, C_0\right)\right\}$$

$$\approx_{(\mathsf{poly}(\lambda), \epsilon(\lambda))} \left\{\mathsf{O}_{C_1} \ : \ \mathsf{O}_{C_1} \leftarrow \mathsf{iO}\left(1^\lambda, 1^s, C_1\right)\right\} \ .$$

Different security levels. For arbitrary functions $f_0, f_1 : \mathbb{N} \to \mathbb{N}$, we say that an iO scheme is $\left(f_0, \frac{1}{f_1}\right)$-secure, if in the above security part of the definition, we swap poly with a specific function f_0 and the negligible function with the function $\frac{1}{f_1}$. Concretely, a sub-exponentially secure iO scheme would be one such that there exists a positive real constant $c > 0$ such that the scheme is $\left(2^{\lambda^c}, \frac{1}{2^{\lambda^c}}\right)$-secure.

Definition 9 (Lossy Functions)**.** *A lossy function (LF) scheme consists of classical algorithms* $(\mathsf{LF.KeyGen}, \mathsf{LF.F})$ *with the following syntax.*

- $\mathsf{pk} \leftarrow \mathsf{LF.KeyGen}\left(1^\lambda, b, 1^\ell\right)$*: a probabilistic polynomial-time algorithm that gets as input the security parameter* $\lambda \in \mathbb{N}$*, a bit* b *and a lossyness parameter* $\ell \in \mathbb{N}$*,* $\lambda \geq \ell$*. The algorithm outputs a public key.*

- $y \leftarrow \mathsf{LF.F}\left(\mathsf{pk}, x\right)$*: a deterministic polynomial-time algorithm that gets as input the security parameter* $\lambda \in \mathbb{N}$*, the public key* pk *and an input* $x \in \{0,1\}^\lambda$ *and outputs a string* $y \in \{0,1\}^m$ *for some* $m \geq \lambda$*.*

The scheme satisfies the following guarantees.

- **Statistical Correctness for Injective Mode:** *There exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for every* $\lambda, \ell \in \mathbb{N}$*,*

$$\Pr_{\mathsf{pk} \leftarrow \mathsf{LF.KeyGen}\left(1^\lambda, 0, 1^\ell\right)}\left[\left|Img\left(\mathsf{LF.F}\left(\mathsf{pk}, \cdot\right)\right)\right| = 2^\lambda\right] \geq 1 - \mathsf{negl}(\lambda) \ .$$

- **Statistical Correctness for Lossy Mode:** *There exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for every* $\lambda, \ell \in \mathbb{N}$*,*

$$\Pr_{\mathsf{pk} \leftarrow \mathsf{LF.KeyGen}\left(1^\lambda, 1, 1^\ell\right)}\left[\left|Img\left(\mathsf{LF.F}\left(\mathsf{pk}, \cdot\right)\right)\right| \leq 2^\ell\right] \geq 1 - \mathsf{negl}(\lambda) \ .$$

- **Security:** *For every polynomial* $\mathsf{poly}(\cdot)$ *there exists a negligible function* ϵ *such that the following holds. Let* $\lambda, \ell \in \mathbb{N}$*, then (note that in the following computational indistinguishability, the security parameter is* ℓ *and not* λ*),*

$$\left\{\mathsf{pk}_0 \ : \ \mathsf{pk}_0 \leftarrow \mathsf{LF.KeyGen}\left(1^\lambda, 0, \ell\right)\right\}$$

$$\approx_{(\mathsf{poly}(\ell), \epsilon(\ell))} \left\{\mathsf{pk}_1 \ : \ \mathsf{pk}_1 \leftarrow \mathsf{LF.KeyGen}\left(1^\lambda, 1, \ell\right)\right\} \ .$$

Different security levels. For arbitrary functions $f_0, f_1 : \mathbb{N} \to \mathbb{N}$, we say that an LF scheme is $\left(f_0, \frac{1}{f_1}\right)$-secure, if in the above security part of the definition, we swap poly with a specific function f_0 and the negligible function with the function $\frac{1}{f_1}$. Concretely, a sub-exponentially secure LF scheme would be one such that there exists a positive real constant $c > 0$ such that the scheme is $\left(2^{\ell^c}, \frac{1}{2^{\ell^c}}\right)$-secure.

2.1 Learning With Errors

Let χ_σ be the distribution over $\mathbb{Z}$ where $\Pr[x \leftarrow \chi_\sigma] \propto e^{2\pi x^2/\sigma^2}$.

Definition 10. *Let* T, p, m, q, σ *be functions in* n *where* m*,* $\log(q)$*, and* $\log(\sigma)$ *are bounded by polynomials in* n*. The* (T, p, m, q, σ)*-LWE assumption holds if, for any* T*-time adversary* $\mathcal{A}$*,*

$$\left|\Pr\left[\mathcal{A}(A, v) = 1 : \substack{A \leftarrow \mathbb{Z}_q^{m \times n} \\ v \leftarrow \mathbb{Z}_q^m}\right] - \Pr\left[\mathcal{A}(A, v) = 1 : \substack{A \leftarrow \mathbb{Z}_q^{m \times n} \\ s \leftarrow \mathbb{Z}_q^n, e \leftarrow \chi_\sigma^m \\ v \leftarrow A \cdot s + e \bmod q}\right]\right| \leq p$$

We say that LWE is $2^{O(\lambda/\log\lambda)}$-secure with a polynomial noise-modulus ratio if the (T, p, m, q, σ)-LWE assumption holds for $\sigma = n^{O(1)}$, $q = \sigma \times n^{O(1)}$, $m = \Omega(n \log q)$, and for any $T/p \leq 2^{O(n)}$. Here, we are implicitly setting $\lambda = O(n \log q) = O(n \log n)$, which corresponds to the bit-length of the LWE secret vector s.

We say that LWE is optimally-secure if the (T, p, m, q, σ)-LWE holds for some $\sigma = n^{O(1)}$, $q = 2^{O(n)}$, and $m = \Omega(n \log q)$, and for any $T/p \leq 2^{O(n)}$.

Both of these assumptions appear consistent with known lattice attacks, with optimal LWE being the stronger of the two. For optimal LWE in particular, attacks running in time sub-exponential in n seen to require $q/\sigma \geq 2^{\Omega(\sqrt{n \log q})}$, which in the regime of polynomial σ translates to $q \geq 2^{\Omega(n)}$. By setting $q = 2^{O(n)}$ where the constant in the $O(n)$ is sufficiently small, we hope to avoid such attacks.

Lossy functions from LWE. In this paper, we will need a lossy function that is $(f_0, 1/f_1)$-secure for $f_0 = f_1 = 2^{\ell/\mathsf{polylog}\ell}$. We briefly discuss how to achieve this from our strong variants of LWE.

In [WZ24], building on prior works [PW08, AKPW13], lossy functions are considered from LWE. We will not give the whole construction here, but just remark that in their construction, they are able to obtain ℓ set to be proportional to the bit-length of the LWE secret, namely $\ell = n \log q$. They only need polynomial noise and modulus. As such, by the $2^{O(\lambda/\log\lambda)}$-secure LWE assumption, we can set $n = O(\lambda)$ and hence $\ell = O(\lambda \log \lambda)$, giving the desired lossiness:

Theorem 11 (Implicit in [WZ24])**.** *Assuming LWE is $2^{O(\lambda/\log\lambda)}$-secure, there exists lossy functions that are $(2^{O(\lambda/\log\lambda)}, 2^{-O(\lambda/\log\lambda)})$-secure.*

2.2 Two iO Techniques

Here, we recall two standard iO techniques that we will abstract as useful lemmas.

Sparse Random Triggers. Let P be some program and P' an arbitrary different program. Let R be a function with range $[N]$ for some N that is exponential in the security parameter. Let J_y (for 'join') be the following program:

$$J_y(x) = \begin{cases} P(x) \text{ if } R(x) \neq y \\ P'(x) \text{ if } R(x) = y \end{cases} .$$

Lemma 12. *Suppose one-way functions exist. For sufficiently large polynomial s and for y chosen uniformly in $\{0,1\}^\lambda$, $\mathsf{iO}(1^\lambda, 1^s, P)$ and $\mathsf{iO}(1^\lambda, 1^s, J_y)$ are computationally indisitnguishable even given the description of P. Moreover, y is computationally unpredictable given $P, \mathsf{iO}(1^\lambda, 1^s, J_y)$*

Swapping distributions. We now move to the next standard technique. Let $\{D_0^x\}_x, \{D_1^x\}_x$ be two families of distributions over the same domain $\mathcal{Y}$, which can also be thought of as deterministic functions $D_0(x; r)$, $D_1(x; r)$ that take as input an index x and some random coins r. Let P be a program that makes queries to an oracle $O : \mathcal{X} \to \mathcal{Y}$ for some set $\mathcal{X}$. Then we have the following:

Lemma 13. *Let $(\mathsf{F}, \mathsf{Punc})$ be a $(f_\mathsf{F}, \delta_\mathsf{F})$-secure puncturable PRF and iO be a $(f_\mathsf{iO}, \delta_\mathsf{iO})$-secure iO. Let $\mathcal{X}$ a finite set and let $D_0 := \{D_{0,x}\}_{x\in\mathcal{X}}, D_1 := \{D_{1,x}\}_{x\in\mathcal{X}}$ two ensembles of distributions, such that for every $x \in \mathcal{X}$, $D_{0,x}$, $D_{1,x}$ are (f_D, δ_D)-indistinguishable. Let $E_0(x) = D_0(x; \mathsf{F}(k, x))$ and $E_1(x) =$*

$D_1\left(x; \mathsf{F}\left(\overline{k}, x\right)\right)$. *Then for a sufficiently large polynomial* s, $\mathsf{iO}\left(1^\lambda, 1^s, P^{E_0}\right)$ *and* $\mathsf{iO}\left(1^\lambda, 1^s, P^{E_1}\right)$ *are* $\left(\min\left(f_{\mathsf{F}}, f_{\mathsf{iO}}, f_D\right), O\left(|\mathcal{X}| \cdot \left(\delta_{\mathsf{F}} + \delta_{\mathsf{iO}} + \delta_D\right)\right)\right)$*-computationally indistinguishable, where* $k, \overline{k} \leftarrow \{0,1\}^\lambda$ *are uniformly random keys.*

2.3 Subspace Hiding Functions

We prove a generalization of subspace-hiding obfuscation, which we call subspace-hiding function. Basically, the idea is this: We have a small subspace S_0, and a (possibly significantly) bigger subspace S, which contains S_0. Then, we have a partition P of the bigger space S into cosets, all of them have the same size and all are parallel cosets to S_0 (that is, each of the cosets in the partition P is in the form of $x + S_0$ for some $x \in S$). Then, there is access to the function F_S which given $\mathbf{v} \in \mathbb{Z}_2^k$, checks whether it is in S and if it does then it outputs the index of the coset it belongs to inside the partition P.

The statement intuitively shows that F hides S_0 as long as the dimension of S (and thus in particular the dimension of S_0) isn't too big. The proof extends the proof of subspace hiding from [Zha19] and generalizes the statement. We also allow to get ℓ-duplicated the access to F (for a general ℓ), which is sometimes helpful for using the statement as a black box. The formal statement follow.

Lemma 14 (Subspace Hiding Function)**.** *Let* $k, r, s, \lambda \in \mathbb{N}$ *such that* $r + \lambda + s \leq k$*, let* $S \subseteq \mathbb{Z}_2^k$ *a subspace of dimension* $r + \lambda$ *and let* $S_0 \subset S$ *a subspace of dimension* r*. Let* P *any partition of* S*, the bigger space, into* 2^λ *cosets that are all parallel to* S_0*, the smaller space. Let* $F_S : \mathbb{Z}_2^k \to \left(\{\bot, \top\} \times \mathbb{Z}_2^\lambda\right)$ *be the function that given input* $\mathbf{v} \in \mathbb{Z}_2^k$*, first checks whether* $\mathbf{v} \in S$ *and outputs* $\bot$ *if not, and otherwise outputs the index* $\mathbf{c}_\mathbf{v} \in \mathbb{Z}_2^\lambda$ *of the coset of* S_0 *(inside the big space* S*), such that* $\mathbf{v}$ *belongs to it.*

Let $\mathcal{S}_s$ *the uniform distribution over subspaces* T_0 *of dimension* $r + s$ *such that* $S_0 \subset T_0 \subset \mathbb{Z}_2^k$*. We define* T *as the linear span of* T_0 *and* S *and define* F_T *as the function which computes the index of parallel cosets of* T_0 *inside* T*, with respect to the same partition* P*. More formally, let* $S_{0,\mathbf{c}}$ *the* $\mathbf{c}$*-th parallel coset of* S_0 *inside the space* S*, for* $\mathbf{c} \in \mathbb{Z}_2^\lambda$*. Let* $T_{0,\mathbf{c}}$ *the linear sum of* T_0 *and* $S_{0,\mathbf{c}}$*. Then* $T_{0,\mathbf{c}}$ *is the* $\mathbf{c}$*-th parallel coset of* T_0 *inside the space* T*. Then, we have the following guarantees.*

Information theoretical indistinguishability in the oracle model: *For every* $\ell \in \mathbb{N}$*, for every oracle-aided quantum algorithm* $\mathcal{A}$ *making at most* q *quantum queries, we have the following indistinguishability between oracle distributions:*

$$\{\mathcal{O}_{F_S}^{\otimes \ell}\} \approx_{\frac{\lambda \cdot \ell \cdot q \cdot s}{\sqrt{2^{k-r-\lambda-s}}}} \{\mathcal{O}_{F_T}^{\otimes \ell} \ : \ T_0 \leftarrow \mathcal{S}_s\} \ .$$

Computational indistinguishability in the standard model: *Denote* $\lambda' := \lambda_{\mathsf{OWF}} := k - r - \lambda - s$*,* $\lambda_{\mathsf{iO}} := \mathsf{poly}(\lambda')$ *security parameters for OWF and iO, and let* iO *an indistinguishability obfuscation scheme that is* $(f_{\mathsf{iO}}(\cdot), \epsilon_{\mathsf{iO}}(\cdot))$*-secure, and assume that* $(f_{\mathsf{OWF}}(\cdot), \epsilon_{\mathsf{OWF}}(\cdot))$*-secure injective one-way functions exist. Then, for a sufficiently large* $p := p(\lambda')$ *polynomial in the security parameter, we have the following indistinguishability for* $f(\lambda') := \min\left(f_{\mathsf{OWF}}(\lambda'), f_{\mathsf{iO}}\left(\lambda_{\mathsf{iO}}\right)\right)$*,* $\epsilon(\lambda') := \max\left(\epsilon_{\mathsf{OWF}}(\lambda'), \epsilon_{\mathsf{iO}}\left(\lambda_{\mathsf{iO}}\right)\right)$ *and for every* $\ell \in \mathbb{N}$

$$\{\mathsf{O}_{F_S} \ : \ \mathsf{O}_{F_S} \leftarrow \mathsf{iO}\left(1^{\lambda_{\mathsf{iO}}}, 1^p, C_{F_S}\right)\}^{\otimes \ell} \approx_{\left(\frac{f(\lambda') - \mathsf{poly}(\lambda')}{\lambda \cdot \ell}, \ s \cdot \epsilon(\lambda')\right)}$$

$$\{\{\mathsf{O}_{F_T} \ : \ \mathsf{O}_{F_T} \leftarrow \mathsf{iO}\left(1^{\lambda_{\mathsf{iO}}}, 1^p, C_{F_T}\right)\}^{\otimes \ell} \ : \ T_0 \leftarrow \mathcal{S}_s\} \ .$$

Proof. We prove the claim by a hybrid argument with s steps, increasing the dimension of the random superspace T_0 by 1 at each step.

We start with algebraically characterizing the subspaces and cosets involved in the proof. Specifically, the goal of the first step of the proof will be to reduce the task of indistinguishability between the functions to indistinguishability in the form of subspace hiding. Observe that for the subspaces S_0, S and the partition P of parallel cosets to S_0, there exists a matrix $\mathbf{S} \in \mathbb{Z}_2^{k\times(k-r)}$ such that

$$S_0 = \mathsf{ColSpan}\left(\mathbf{S}\right)^{\perp} \quad , \quad S = \mathsf{ColSpan}\left(\mathbf{S}^{(k-r-\lambda)}\right)^{\perp} \quad ,$$

where $\mathbf{S}^{(k-r-\lambda)} \in \mathbb{Z}_2^{k\times(k-r-\lambda)}$ are the $k-r-\lambda$ leftmost columns of $\mathbf{S}$. Also, $\mathbf{S}$ satisfies that for the partition P and every input $\mathbf{v} \in \mathbb{Z}_2^k$ to F, the output can be computed using $\mathbf{S}$ as follows: check that $\mathbf{v}^T \cdot \mathbf{S}^{(k-r-\lambda)} = 0^{k-r-\lambda}$ and output $\perp$ if not. Otherwise, the output is the inner product between $\mathbf{v}$ and the remaining λ rightmost columns of $\mathbf{S}$.

We next define λ subspaces as follows: For $j \in [\lambda]$ we have $S_j = \mathsf{ColSpan}\left(\mathbf{S}^{(k-r-\lambda)}, \mathbf{S}_j\right)^{\perp}$, where $\mathbf{S}_j$ is j-th rightmost column of $\mathbf{S}$. So, for every $j \in [\lambda]$, the subspace S_j is the set of vectors $\mathbf{v} \in \mathbb{Z}_2^k$ such that $\mathbf{v}^T \cdot \left(\mathbf{S}^{(k-r-\lambda)}, \mathbf{S}_j\right) = 0^{k-r-\lambda+1}$.

We next describe a further equivalent way for computing F using only black-box access to the membership checks for S and the λ additional subspaces: First check that the input vector $\mathbf{v}$ is in S and output $\perp$ if not. Then, the j-th output bit of F is exactly the negation of the membership of $\mathbf{v}$ inside S_j. So, at this point we described the computation of F using black-box access to a small number of subspaces on the one hand, but also there's a matrix dominating this computation, inside the black boxes of the membership checks.

Increasing the dimension of all subspaces all at once. Consider the sequence of subspaces $\left(S_0^0, \cdots, S_\lambda^0, S^0\right)$ defined as $(S_0, \cdots, S_\lambda, S)$, and we will increase the dimension of all $\lambda+2$ subspaces simultaneously by 1, indistinguishably and randomly. We do this as follows: we sample a uniformly random $\mathbf{a} \in \mathbb{Z}_2^{k-r}$ and consider the modified subspaces $S_0^1, \cdots, S_\lambda^1, S^1$ such that for $j \in [\lambda]$, the subspace S_j^1 is the set of vectors $\mathbf{v} \in \mathbb{Z}_2^k$ such that $\mathbf{v}^T \cdot \left(\mathbf{S}^{(k-r-\lambda)}, \mathbf{S}_j\right)$ either equals $0^{k-r-\lambda+1}$ or to the corresponding elements of $\mathbf{a}$: Formally, the leftmost $k-r-\lambda$ elements of $\mathbf{a}$ should equal $\mathbf{v}^T \cdot \mathbf{S}^{(k-r-\lambda)}$ and also we should have the equality $\mathbf{a}_j = \mathbf{v}^T \cdot \mathbf{S}_j$, where $\mathbf{a}_j$ is the j-th rightmost element of $\mathbf{a}$. Similarly, the subspace S_0^1 is the set of vectors $\mathbf{v} \in \mathbb{Z}_2^k$ such that $\mathbf{v}^T \cdot \mathbf{S}$ either equals 0^{k-r} or to the entirety of $\mathbf{a}$. The subspace S^1 is the set of vectors $\mathbf{v} \in \mathbb{Z}_2^k$ such that $\mathbf{v}^T \cdot \mathbf{S}^{(k-r-\lambda)}$ either equals $0^{k-r-\lambda}$ or to the corresponding leftmost elements of $\mathbf{a}$.

We have the following guarantees:

- For all $j \in [\lambda]$, conditioned on the leftmost $k-r-\lambda$ elements of $\mathbf{a}$ do not equal $0^{k-r-\lambda}$ (which happens with overwhelming probability), the subspace S_j^1 is a random superspace of S_j^0, with one more dimension.

- **Simulation of all other membership checks $S_0^b, \cdots, S_\lambda^b$ using oracle access to the biggest subspace S^b:** We consider oracle access to $\mathcal{O}$, which is a membership check for S^b, for either $b=0$ or $b=1$. We use it to implement all other $\lambda+1$ membership checks $S_0^b, \cdots, S_\lambda^b$. We are given oracle access to membership check in S^b and recall that it is given by the inner product with the matrix $\mathbf{S}^{(k-r-\lambda)}$ and the last $k-r-\lambda$ elements of the uniformly random $\mathbf{a}$. We first sample the remaining λ elements of $\mathbf{a}$ ourselves. In order to implement oracle access

to S_0^b for input $\mathbf{v} \in \mathbb{Z}_2^k$, we check membership in S^b, if the answer was negative we reject. If the answer was positive, we check whether $\mathbf{v}^T \cdot \mathbf{S}^{(k-r-\lambda)} = 0^{k-r-\lambda}$: if so, we accept iff $\mathbf{v}^T \cdot \mathbf{S}_j = 0$, and if $\mathbf{v}^T \cdot \mathbf{S}^{(k-r-\lambda)} \neq 0^{k-r-\lambda}$, we accept iff $\mathbf{v}^T \cdot \mathbf{S}_j$ equals the element we sampled (ourselves) for the j-th element of $\mathbf{a}$. The simulation of the smallest membership check S_0^b is done analogously (by checking all λ columns instead of a single one). One can verify that for $b \in \{0,1\}$, if the oracle access was for S^b then we simulated access for $S_0^b, \cdots, S_\lambda^b$. this means that overall a parallel query to all $\lambda+1$ oracles can be simulated by $\lambda+2$ oracle queries to the biggest membership check. We obtain the following indistinguishability statements given this simulation.

- **Information theoretical indistinguishability:** Due to the randomness of $\mathbf{a}$ (and in particular its leftmost $k-r-\lambda$ elements), by standard quantum lower bounds for unstructured search, for every $\ell \in \mathbb{N}$ we have that for any q-query algorithm,

$$\{\left(\mathcal{O}_{S_j^0}\right)_{j\in\{0,1,\cdots,\lambda\}}, \mathcal{O}_{S^0}\}^{\otimes\ell} \approx_{\frac{\lambda\cdot\ell\cdot q}{\sqrt{2^{k-r-\lambda}}}} \{\left(\mathcal{O}_{S_j^1}\right)_{j\in\{0,1,\cdots,\lambda\}}, \mathcal{O}_{S^1}\}^{\otimes\ell} \ .$$

- **Computational indistinguishability:** We can think of the leftmost $k-r-\lambda$ bits of the uniformly random $\mathbf{a} \in \mathbb{Z}_2^{k-r}$ as a secret image of an injective one-way function. By known techniques in indistinguishability obfuscation and one-way functions (subspace-hiding obfuscation [Zha21b] and extractability obfuscation [BCP14]) we have,

$$\{\mathsf{O}_{S_0^0}, \cdots, \mathsf{O}_{S_\lambda^0}, \mathsf{O}_{S^0} \ : \ \mathsf{O}_{S_i^0} \leftarrow \mathsf{iO}\left(1^{\lambda_{\mathsf{iO}}}, 1^p, C_{S_i^0}\right)\}^{\otimes\ell} \approx_{\left(\frac{f(\lambda')-\mathsf{poly}(\lambda')}{\lambda\cdot\ell}, \epsilon(\lambda')\right)}$$

$$\{\{\mathsf{O}_{S_0^1}, \cdots, \mathsf{O}_{S_\lambda^1}, \mathsf{O}_{S^1} \ : \ \mathsf{O}_{S_i^1} \leftarrow \mathsf{iO}\left(1^{\lambda_{\mathsf{iO}}}, 1^p, C_{S_i^1}\right)\}^{\otimes\ell} \ : \ \mathbf{a} \leftarrow \mathbb{Z}_2^{k-r}\}$$

Finally, note that if we compute F using the (indistinguishable) membership checks for $S_0^1, \cdots, S_\lambda^1, S^1$ instead of the original $S_0^0, \cdots, S_\lambda^0, S^0$, we get a corresponding partition and specifically, we get exactly a function F_T such that the random superspace T_0 of S_0 is one dimension bigger.

Generalizing to an arbitrary s. The above proves our claim for $s = 1$. For a general s we continue making dimension additions in the exact same manner. At each of the steps $i \in [s]$ we have the subspaces $\{S_j^{i-1}\}_{j\in\{0,1,\cdots,\lambda\}}, S^{i-1}$ and we move to subspaces $\{S_j^i\}_{j\in\{0,1,\cdots,\lambda\}}, S^i$ that are one dimension bigger. At the beginning of every step i we also re-initialize the matrix for a different size $\mathbf{S}_i \in \mathbb{Z}_2^{k\times(k-r-(i-1))}$. Note that at step $i \in [s]$ the oracle indistinguishability is (for any q-query algorithm),

$$\{\{\mathcal{O}_{S_j^{i-1}}\}_{j\in\{0,1,\cdots,\lambda\}}, \mathcal{O}_{S^{i-1}}\}^{\otimes\ell} \approx_{\frac{\lambda\cdot\ell\cdot q}{\sqrt{2^{k-r-\lambda-(i-1)}}}} \{\{\mathcal{O}_{S_j^i}\}_{j\in\{0,1,\cdots,\lambda\}}, \mathcal{O}_{S^i}\}^{\otimes\ell} \ ,$$

and the computational indistinguishability is

$$\{\mathsf{O}_{S_0^{i-1}}, \cdots, \mathsf{O}_{S_\lambda^{i-1}}, \mathsf{O}_{S^{i-1}} \ : \ \mathsf{O}_{S_j^{i-1}} \leftarrow \mathsf{iO}\left(1^{\lambda_{\mathsf{iO}}}, 1^p, C_{S_j^{i-1}}\right)\}^{\otimes\ell} \approx_{\left(\frac{f(\lambda')-\mathsf{poly}(\lambda')}{\lambda\cdot\ell}, \epsilon(\lambda')\right)}$$

$$\{\{\mathsf{O}_{S_0^i}, \cdots, \mathsf{O}_{S_\lambda^i}, \mathsf{O}_{S^i} \ : \ \mathsf{O}_{S_j^i} \leftarrow \mathsf{iO}\left(1^{\lambda_{\mathsf{iO}}}, 1^p, C_{S_j^i}\right)\}^{\otimes\ell} \ : \ \mathbf{a} \leftarrow \mathbb{Z}_2^{k-r}\} \ .$$

Overall we get that for a q-query algorithm $\mathcal{A}$, the distinguishing advantage between

$$\{\mathcal{O}_{S_0},\cdots,\mathcal{O}_{S_\lambda},\mathcal{O}_S\}^{\otimes\ell}\ ,\ \{\mathcal{O}_{T_0},\cdots,\mathcal{O}_{T_\lambda},\mathcal{O}_T\ :\ T_0\leftarrow\mathcal{S}_s\}^{\otimes\ell}$$

is

$$\sum_{i\in\{0,1,\cdots,s-1\}}\frac{\lambda\cdot\ell\cdot q}{\sqrt{2^{k-r-\lambda-i}}}\leq\frac{\lambda\cdot\ell\cdot q\cdot s}{\sqrt{2^{k-r-\lambda-s}}}\ ,$$

as needed for the oracle case.

As for the computational case, we have a similar indistinguishability

$$\{\mathsf{O}_{S_0},\cdots,\mathsf{O}_{S_\lambda},\mathsf{O}_S\ :\ \mathsf{O}_{S_j}\leftarrow\mathsf{iO}\left(1^{\lambda_{\mathsf{iO}}},1^p,C_{S_j}\right)\}^{\otimes\ell}\approx_{\left(\frac{f(\lambda')-\mathsf{poly}(\lambda')}{\lambda\cdot\ell},\ s\cdot\epsilon(\lambda')\right)}$$

$$\{\{\mathsf{O}_{T_0},\cdots,\mathsf{O}_{T_\lambda},\mathsf{O}_T\ :\ T\leftarrow\mathcal{S}_s,\mathsf{O}_{T_j}\leftarrow\mathsf{iO}\left(1^{\lambda_{\mathsf{iO}}},1^p,C_{T_j}\right)\}^{\otimes\ell}\ :\ T_0\leftarrow\mathcal{S}_s\}\ .$$

To end the proof, it remains to observe that in the same manner that we implemented F_S using membership checks for the subspaces $S_0,S_1,\cdots,S_\lambda,S$, we can use the membership checks for $T_0,T_1,\cdots,T_\lambda,T$ to implement the function F_T. This finishes our proof. □

Indistinguishability between static and independent repetitions of F_T. We next state the following Corollary below, which will be useful for followup supporting lemmas. The Corollary intuitively says that the following two distributions are indistinguishable. The first is where we sample $T_0\leftarrow\mathcal{S}_s$ once and then give ℓ-duplicate access to F_T (in the case of standard-model obfuscated circuits, this means we sample ℓ i.i.d. obfuscations of F_T). The second distribution is where we generate ℓ i.i.d. samples, that is, for every $i\in[\ell]$ we sample a fresh $T_0^i\leftarrow\mathcal{S}_s$, and give access to the ℓ functions $F_{T^1},\cdots,F_{T^\ell}$. The formal statement and proof follow.

Corollary 15. *[Indistinguishability between static and independent repetitions of F_T] Let $k,r,s,\lambda\in\mathbb{N}$ such that $r+\lambda+s\leq k$, let $S\subseteq\mathbb{Z}_2^k$ a subspace of dimension $r+\lambda$ and let $S_0\subset S$ a subspace of dimension r. Let P any partition of S, the bigger space, into 2^λ cosets that are all parallel to S_0, the smaller space. Let $F_S:\mathbb{Z}_2^k\to\left(\{\bot,\top\}\times\mathbb{Z}_2^\lambda\right)$ be the function that given input $\mathbf{v}\in\mathbb{Z}_2^k$, first checks whether $\mathbf{v}\in S$ and outputs $\bot$ if not, and otherwise outputs the index $\mathbf{c}_\mathbf{v}\in\mathbb{Z}_2^\lambda$ of the coset of S_0 (inside the big space S), such that $\mathbf{v}$ belongs to it.*

Let $\mathcal{S}_s$ the uniform distribution over subspaces T_0 of dimension $r+s$ such that $S_0\subset T_0\subset\mathbb{Z}_2^k$. We define T as the linear span of T_0 and S and define F_T as the function which computes the index of parallel cosets of T_0 inside T, with respect to the same partition P. More formally, let $S_{0,\mathbf{c}}$ the $\mathbf{c}$-th parallel coset of S_0 inside the space S, for $\mathbf{c}\in\mathbb{Z}_2^\lambda$. Let $T_{0,\mathbf{c}}$ the linear sum of T_0 and $S_{0,\mathbf{c}}$. Then $T_{0,\mathbf{c}}$ is the $\mathbf{c}$-th parallel coset of T_0 inside the space T. Then, we have the following guarantees.

***Information theoretical indistinguishability in the oracle model:** For every $\ell\in\mathbb{N}$, for every oracle-aided quantum algorithm $\mathcal{A}$ making at most q quantum queries, we have the following indistinguishability between oracle distributions:*

$$\{\mathcal{O}_{F_T}^{\otimes\ell}\ :\ T_0\leftarrow\mathcal{S}_s\}\approx_{\frac{2\cdot\lambda\cdot\ell\cdot q\cdot s}{\sqrt{2^{k-r-\lambda-s}}}}\{\mathcal{O}_{F_{T^1}},\cdots,\mathcal{O}_{F_{T^\ell}}\ :\ \forall i\in[\ell],T_0^i\leftarrow\mathcal{S}_s\}\ .$$

***Computational indistinguishability in the standard model:** Denote $\lambda':=\lambda_{\mathsf{OWF}}:=k-r-\lambda-s$, $\lambda_{\mathsf{iO}}:=\mathsf{poly}(\lambda')$ security parameters for OWF and iO, and let iO an indistinguishability obfuscation scheme that is $(f_{\mathsf{iO}}(\cdot),\epsilon_{\mathsf{iO}}(\cdot))$-secure, and assume that $(f_{\mathsf{OWF}}(\cdot),\epsilon_{\mathsf{OWF}}(\cdot))$-secure*

injective one-way functions exist. Then, for a sufficiently large $p := p(\lambda')$ polynomial in the security parameter, we have the following indistinguishability for $f(\lambda') := \min\left(f_{\mathsf{OWF}}(\lambda'), f_{\mathsf{iO}}\left(\lambda_{\mathsf{iO}}\right)\right)$, $\epsilon(\lambda') := \max\left(\epsilon_{\mathsf{OWF}}(\lambda'), \epsilon_{\mathsf{iO}}\left(\lambda_{\mathsf{iO}}\right)\right)$ and for every $\ell \in \mathbb{N}$

$$\left\{\left\{\mathsf{O}_{F_T} \ : \ \mathsf{O}_{F_T} \leftarrow \mathsf{iO}\left(1^{\lambda_{\mathsf{iO}}}, 1^p, C_{F_T}\right)\right\}^{\otimes \ell} \ : \ T_0 \leftarrow \mathcal{S}_s\right\} \approx_{\left(\frac{f(\lambda')-\mathsf{poly}(\lambda')}{2\cdot\lambda\cdot\ell},\, s\cdot\epsilon(\lambda')\right)}$$

$$\left\{\mathsf{O}_{F_{T^i}}, \cdots, \mathsf{O}_{F_{T^\ell}} \ : \ \forall i \in [\ell], T_0^i \leftarrow \mathcal{S}_s, \mathsf{O}_{F_{T^i}} \leftarrow \mathsf{iO}\left(1^{\lambda_{\mathsf{iO}}}, 1^p, C_{F_{T^i}}\right)\right\} \ .$$

Proof. The proof is simple and is given by two steps. We start with the distribution on the left, whether it is the oracle model case or standard-model obfuscations. It is indistinguishable from ℓ-access to just F_S, with parameters given by Lemma 14. Next, in the second step of the proof we do an ℓ-step hybrid argument, invoking the same Lemma 14, each time using the "ℓ" in the Lemma 14 equal 1, but use it ℓ times here. Overall, we get the indistinguishability parameters (in the oracle model or standard model) as in this Corollary 15. □

2.4 Hardness of Concentration in Hidden Random Subspace

We re-prove and generalize results from [SZ25, Shm22b, Shm22a], regarding hardness of finding vectors that are always inside some hidden subspace T. Specifically, we generalize them to the setting including subspace-hiding functions (which generalize subspace-hiding of a single subspace membership check). The formal statement and proof follow.

Lemma 16 (IO Dual Subspace Anti-Concentration)**.** *Let $k, r, s, \lambda \in \mathbb{N}$ such that $r + \lambda + s \leq k$, let $S \subseteq \mathbb{Z}_2^k$ a subspace of dimension $r + \lambda$ and let $S_0 \subset S$ a subspace of dimension r. Let P any partition of S, the bigger space, into 2^λ cosets that are all parallel to S_0, the smaller space. Let $F_S : \mathbb{Z}_2^k \to \left(\{\bot, \top\} \times \mathbb{Z}_2^\lambda\right)$ be the function that given input $\mathbf{v} \in \mathbb{Z}_2^k$, first checks whether $\mathbf{v} \in S$ and outputs $\bot$ if not, and otherwise outputs the index $\mathbf{c_v} \in \mathbb{Z}_2^\lambda$ of the coset of S_0 (inside the big space S), such that $\mathbf{v}$ belongs to it.*

Let $\mathcal{S}_s$ the uniform distribution over subspaces T_0 of dimension $r + s$ such that $S_0 \subset T_0 \subset \mathbb{Z}_2^k$. We define T as the linear span of T_0 and S and define F_T as the function which computes the index of parallel cosets of T_0 inside T, with respect to the same partition P. More formally, let $S_{0,\mathbf{c}}$ the $\mathbf{c}$-th parallel coset of S_0 inside the space S, for $\mathbf{c} \in \mathbb{Z}_2^\lambda$. Let $T_{0,\mathbf{c}}$ the linear sum of T_0 and $S_{0,\mathbf{c}}$. Then $T_{0,\mathbf{c}}$ is the $\mathbf{c}$-th parallel coset of T_0 inside the space T. Then, we have the following guarantees.

Information theoretical hardness in the oracle model: *Assume there is an oracle-aided quantum algorithm $\mathcal{A}$ making at most q quantum queries and outputting a vector $\mathbf{u} \in \mathbb{Z}_2^k$ at the end of its execution, such that*

$$\Pr\left[\mathcal{A}^{\mathcal{O}_{F_T}} \in \left(T_0^\perp \setminus \{0\}\right) \ : \ T_0 \leftarrow \mathcal{S}_s\right] \geq \epsilon \ .$$

Also, denote $t := k - r - s$, $\ell := \frac{k(t+1)}{\epsilon}$ and assume the following conditions:

1. $\frac{t}{2^{s-t}} \leq \frac{\epsilon}{2}$, *and*
2. $\frac{3\cdot\lambda\cdot\ell^2\cdot q\cdot s}{\sqrt{2^{k-r-\lambda-s}}} \leq \frac{1}{2}$.

Then, it is necessarily the case that

$$\Pr\left[\mathcal{A}^{\mathcal{O}_{F_T}} \in \left(S_0^{\perp} \setminus T_0^{\perp}\right) \ : \ T_0 \leftarrow \mathcal{S}_s\right] \geq \frac{\epsilon}{16 \cdot k \cdot (t+1)} \ .$$

Computational hardness in the standard model: *Denote* $\lambda' := \lambda_{\mathsf{OWF}} := k-r-\lambda-s$, $\lambda_{\mathsf{iO}} := \mathsf{poly}(\lambda')$ *security parameters for OWF and iO, and let* iO *an indistinguishability obfuscation scheme that is* $(f_{\mathsf{iO}}(\cdot), \epsilon_{\mathsf{iO}}(\cdot))$*-secure, and assume that* $(f_{\mathsf{OWF}}(\cdot), \epsilon_{\mathsf{OWF}}(\cdot))$*-secure injective one-way functions exist. Then, for a sufficiently large* $p := p(\lambda')$ *polynomial in the security parameter, we have the following guarantees for* $f(\lambda') := \min\left(f_{\mathsf{OWF}}(\lambda'), f_{\mathsf{iO}}\left(\lambda_{\mathsf{iO}}\right)\right)$, $\epsilon(\lambda') := \max\left(\epsilon_{\mathsf{OWF}}(\lambda'), \epsilon_{\mathsf{iO}}\left(\lambda_{\mathsf{iO}}\right)\right)$. *Denote by* $\mathcal{O}_{\lambda,p,s}$ *the distribution over obfuscated circuits that samples* $T_0 \leftarrow \mathcal{S}_s$ *and then* $\mathsf{O}_{F_T} \leftarrow \mathsf{iO}\left(1^{\lambda_{\mathsf{iO}}}, 1^p, C_{F_T}\right)$.

Assume there is a quantum algorithm $\mathcal{A}$ *of complexity* $T_{\mathcal{A}}$ *such that,*

$$\Pr\left[\mathcal{A}\left(\mathsf{O}_{F_T}\right) \in \left(T_0^{\perp} \setminus \{0\}\right) \ : \ \mathsf{O}_{F_T} \leftarrow \mathcal{O}_{\lambda,p,s}\right] \geq \epsilon \ .$$

Also, denote $t := k-r-s$, $\ell := \frac{k(t+1)}{\epsilon}$ *and assume the following conditions:*

1. $\frac{t}{2^{s-t}} \leq \frac{\epsilon}{2}$, *and*
2. $\ell \cdot T_{\mathcal{A}} + \ell \cdot k^3 \leq \frac{f(\lambda') - \mathsf{poly}(\lambda')}{2 \cdot \lambda \cdot \ell}$, *and*
3. $(s+1) \cdot \epsilon(\lambda') \leq 1/2$.

Then, it is necessarily the case that

$$\Pr\left[\mathcal{A}\left(\mathsf{O}_{F_T}\right) \in \left(S_0^{\perp} \setminus T_0^{\perp}\right) \ : \ \mathsf{O}_{F_T} \leftarrow \mathcal{O}_{\lambda,p,s}\right] \geq \frac{\epsilon}{16 \cdot k \cdot (t+1)} \ .$$

Proof. We start with defining the following reduction $\mathcal{B}$, that will use the circuit $\mathcal{A}$ as part of its machinery.

The reduction $\mathcal{B}$. The input to $\mathcal{B}$ contains $\ell := \frac{k \cdot (t+1)}{\epsilon}$ samples of an access to some F, $\left(\mathsf{O}^{(1)}, \cdots, \mathsf{O}^{(\ell)}\right)$, for $t := k-r-s$. These samples can be oracle access (in the classical oracle model) or obfuscated circuits (in the standard model). Given the ℓ samples, execute $\mathcal{A}\left(\mathsf{O}^{(i)}\right)$ for every $i \in [\ell]$ and obtain ℓ vectors $\{u_1, \cdots, u_\ell\}$. Then, take only the vectors $\{v_1, \cdots, v_m\}$ that are inside $S_0^{\perp}$, and then compute the dimension of their span, $D := \dim\left(\mathsf{Span}\left(v_1, \cdots, v_m\right)\right)$. Note the following properties of the reduction:

- In the oracle model, the number of queries that $\mathcal{B}$ makes is $q \cdot \ell$.
- In the standard model, the running time of $\mathcal{B}$ is $\ell \cdot T_{\mathcal{A}} + \ell \cdot k^3$, where $\ell \cdot T_{\mathcal{A}}$ is for producing the ℓ outputs of $\mathcal{A}$ and $\ell \cdot k^3$ is for (naively) executing Gaussian elimination ℓ times, to repeatedly check whether the new vector v_i adds a dimension i.e., whether it is outside of the span $\mathsf{Span}\left(v_1, \cdots, v_{i-1}\right)$ of the previous vectors.

From hereon we will use the notations for the standard model, and these implicitly mean we are giving black-box oracle access in the classical oracle model. We will be careful to make the distinction, when it is needed, between what happens in the oracle model and what happens in the standard model, when the adversary gets the obfuscated circuits themselves.

Executing $\mathcal{B}$ on the distribution $\mathcal{D}_1$. Consider the following distribution $\mathcal{D}_1$: Sample ℓ i.i.d superspaces $T_0^1, \cdots, T_0^\ell$, and for each of them, send a sample of the function F_{T^i}: $\mathsf{O}_{F_{T^1}}, \cdots, \mathsf{O}_{F_{T^\ell}}$. Observe that this is one of the two distributions proven to be indistinguishable in 15. Let us see what happens when we execute $\mathcal{B}$ on a sample from the distribution $\mathcal{D}_1$.

Consider the ℓ vectors $\{u_1, \cdots, u_\ell\}$ obtained by executing $\mathcal{A}$ on each of the input samples. Recall that $\ell := \frac{1}{\epsilon} \cdot k \cdot (t+1)$ and consider a partition of the vectors into $t+1$ consecutive sequences (or buckets), accordingly, each of length $\frac{1}{\epsilon} \cdot k$. In order to show that the probability for the reduction $\mathcal{B}$ to have $D \geq t+1$ is high, we show that with high probability, in each bucket $j \in [t+1]$ there is a vector u_i that's inside the corresponding dual $\left(T_0^i\right)^\perp$, but such that also the intersection between $\left(T_0^i\right)^\perp$ and the linear span of the previous $j-1$ vectors that hit their duals by $\mathcal{A}$, is only the zero vector 0^k. Note that the last condition indeed implies $D \geq t+1$.

For every $i \in [\ell]$ we define the probability p_i. We start with defining it for the indices i from the first bucket, and then proceed to define it recursively for the rest of the buckets. For indices $i \in [\frac{1}{\epsilon} \cdot k]$ in the first bucket, p_i is the probability, over sampling T_0^i and over the execution of $\mathcal{A}$, for the event that given $\mathsf{O}_{F_{T^i}}$, the output of $\mathcal{A}$ is $u_i \in \left(\left(T_0^i\right)^\perp \setminus \{0\}\right)$, and in such case we define the i-th execution as successful. We denote by $u_{(1)}$ the first vector in the first bucket where a successful execution happens (and define $u_{(1)} := \perp$ if no success happened over the entire bucket). For any i inside any bucket $j \in ([t+1] \setminus \{1\})$ other than the first bucket, we define p_i as the probability that (1) as before, $u_i \in \left(\left(T_0^i\right)^\perp \setminus \{0\}\right)$, and also (2) the intersection between $\left(T_0^i\right)^\perp$ and the linear span of the first vectors (from each respective bucket so far) is only $\{0^k\}$, that is,

$$\left(T_0^i\right)^\perp \cap \mathsf{Span}\left(u_{(1)}, \cdots, u_{(j-1)}\right) = \{0^k\} \ .$$

In words, p_i is the probability that the output of the adversary hits the dual subspace, and also the (randomly-sampled) dual does not have a non-trivial intersection with the space of vectors that we aggregated so far as the first to be successful. We prove that with high probability, all $t+1$ buckets have at least one successful execution.

To see this, we define the following probability p' which we show lower bounds p_i, and is defined as follows. First, let $\overline{T}$ any subspace of dimension bounded by t. $p'_{\overline{T}}$ is the probability that (1) when sampling $T_0^\perp$, the intersection of $T_0^\perp$ with $\overline{T}$ is only the zero vector, and also (2) the output of the adversary $\mathcal{A}$ was inside $T_0^\perp$. p' is defined as the minimal probability taken over all possible choices of $\overline{T}$. After one verifies that indeed for every i we have $p' \leq p_i$, it is sufficient to lower bound p'.

Lower bound for the probability p'. The probability p' is for an event that's defined as the logical AND of two events, and as usual, equals the product between the probability p'_0 of the first event (the trivial intersection between the subspaces), times the conditional probability p'_1 of the second event (that $\mathcal{A}$ hits a non-zero vector in the dual $T_0^\perp$), conditioned on the first event.

First we lower bound the probability p'_0 by upper bounding the complement probability, that is, we show that the probability for a non-trivial intersection is small. Consider the random process of choosing a basis the dual $T_0^\perp$. The process of choosing a basis for the dual has t steps, and in each step we choose a random vector in $S_0^\perp$, that's outside the span we aggregated so far (the first vector can be any non-zero vector in $S_0^\perp$, the second needs to be outside of the span of the vector that was picked in the first step, etc). Given a subspace $\overline{T}$ of dimension t, what is the probability for the two subspaces to have only a trivial intersection? It is exactly the sum over $z \in [t]$ (which

we think of as the steps for sampling $T_0^\perp$) of the following event: In the t-step process of choosing a basis for $T_0^\perp$, index z was the first to cause the subspaces to have a non-zero intersection. Recall that for each $z \in [t]$, the probability that z was such first index to cause an intersection, equals the probability that the z-th sampled basis vector for $T_0^\perp$ is a vector that's inside the unified span of $\overline{T}$ and the aggregated span of $T_0^\perp$ so far, after $z-1$ samples. This amounts to the probability

$$\sum_{z\in[t]} \frac{|\overline{T}| \cdot 2^{z-1}}{|S_0^\perp|} = \sum_{z\in[t]} \frac{2^t \cdot 2^{z-1}}{2^{k-r}} = 2^{-s} \cdot \sum_{z\in\{0,1,\cdots,t-1\}} 2^z$$

$$= 2^{-s} \cdot \left(2^t - 1\right) < 2^{t-s} \ .$$

This means that $p'_0 \geq 1 - 2^{t-s}$.

The lower bound for the conditional probability p'_1 is now quite easy: Note that since $\Pr[A|B] \geq \Pr[A] - \Pr[\neg B]$, letting the event A be the event that the adversary $\mathcal{A}$ outputs a vector in the dual $T_0^\perp$ and B the event that $T_0^\perp$ has only a trivial intersection with $\overline{T}$, we get $p'_1 \geq \epsilon - 2^{t-s}$. By Condition 1 in our statement, it follows that $\frac{t}{2^{s-t}} \leq \frac{\epsilon}{2}$ and thus $p'_1 \geq \frac{\epsilon}{2}$. Overall we got $p' := p'_0 \cdot p'_1 \geq \left(1 - 2^{t-s}\right) \cdot \frac{\epsilon}{2} > \frac{\epsilon}{4}$.

Finally, to see why we get an overall high probability for $D \geq t+1$ on a sample from $\mathcal{D}_1$, observe the following. In each bucket there are $\frac{k}{\epsilon}$ attempts and each succeeds with probability at least $\frac{\epsilon}{4}$ (as we just seen) and thus the overall success probability in a bucket is $\geq 1 - e^{-\Omega(k)}$. Accordingly, the probability to succeed at least once in each of the $t+1$ buckets (and thus to satisfy $D \geq t+1$) is $\geq 1 - (t+1) \cdot e^{-\Omega(k)}$, by considering the complement probability and applying union bound. Overall, when executing $\mathcal{B}$ on a sample from $\mathcal{D}_1$, the probability for $D \geq t+1$ is $\geq 1 - e^{-\Omega(k)}$, in both the oracle model and the standard model using obfuscations.

Executing $\mathcal{B}$ on the distribution $\mathcal{D}_2$. Consider a different distribution $\mathcal{D}_2$: Sample T_0 once, then sample ℓ i.i.d. access to the same function C_{F_T}, that is (in the standard model) $\mathsf{O}_{F_T}^{(1)}, \cdots, \mathsf{O}_{F_T}^{(\ell)}$. By Corollary 15 we have the following indistinguishabilities between $\mathcal{D}_1$ and $\mathcal{D}_2$, in the oracle and standard model.

- In the oracle model the reduction $\mathcal{B}$ executes $q \cdot \ell$ queries, so we have the indistinguishability,

$$\{\mathcal{O}_{F_{T^1}}, \cdots, \mathcal{O}_{F_{T^\ell}} \ : \ \forall i \in [\ell], T_0^i \leftarrow \mathcal{S}_s\} \approx_{\frac{2\cdot\lambda\cdot\ell^2\cdot q\cdot s}{\sqrt{2^{k-r-\lambda-s}}}} \{\mathcal{O}_{F_T}^{\otimes\ell} \ : \ T_0 \leftarrow \mathcal{S}_s\} \ .$$

  Since given a sample from $\mathcal{D}_1$, the algorithm $\mathcal{B}$ outputs $D \geq t+1$ with probability $\geq 1 - e^{-\Omega(k)}$, by the above indistinguishability, whenever we execute $\mathcal{B}$ on a sample from $\mathcal{D}_2$, then with probability at least

$$\geq 1 - e^{-\Omega(k)} - \frac{2 \cdot \lambda \cdot \ell^2 \cdot q \cdot s}{\sqrt{2^{k-r-\lambda-s}}} \geq 1 - \frac{3 \cdot \lambda \cdot \ell^2 \cdot q \cdot s}{\sqrt{2^{k-r-\lambda-s}}}$$

  we have $D \geq t+1$. By our Condition 2 in the Lemma (for the oracle-model case) that $\frac{3\cdot\lambda\cdot\ell^2\cdot q\cdot s}{\sqrt{2^{k-r-\lambda-s}}} \leq \frac{1}{2}$, with probability at least $\frac{1}{2}$ we have $D \geq t+1$ given a sample from $\mathcal{D}_2$, in the oracle model.

- In the standard model, we have the indistinguishability

$$\{\mathsf{O}_{F_{T^i}},\cdots,\mathsf{O}_{F_{T^\ell}} \ : \ \forall i\in[\ell], T_0^i \leftarrow \mathcal{S}_s, \mathsf{O}_{F_{T^i}} \leftarrow \mathsf{iO}\left(1^{\lambda_{\mathsf{iO}}}, 1^p, C_{F_{T^i}}\right)\} \approx_{\left(\frac{f(\lambda')-\mathsf{poly}(\lambda')}{2\cdot\lambda\cdot\ell},\ s\cdot\epsilon(\lambda')\right)}$$

$$\{\{\mathsf{O}_{F_T} \ : \ \mathsf{O}_{F_T} \leftarrow \mathsf{iO}\left(1^{\lambda_{\mathsf{iO}}}, 1^p, C_{F_T}\right)\}^{\otimes\ell} \ : \ T_0 \leftarrow \mathcal{S}_s\} \ .$$

  Recall that the running time of $\mathcal{B}$ is $\ell\cdot T_{\mathcal{A}}+\ell\cdot k^3$ and by our Lemma's Condition 2, this (abovementioned) complexity of $\mathcal{B}$ is $\leq \frac{f(\lambda')-\mathsf{poly}(\lambda')}{2\cdot\lambda\cdot\ell}$. Since given a sample oracle from $\mathcal{D}_1$, the algorithm $\mathcal{B}$ outputs $D\geq t+1$ with probability $\geq 1-e^{-\Omega(k)}$, by the above indistinguishability, whenever we execute $\mathcal{B}$ on a sample from $\mathcal{D}_2$, then with probability at least

$$\geq 1-e^{-\Omega(k)}-s\cdot\epsilon(\lambda')\geq 1-(s+1)\cdot\epsilon(\lambda')$$

  we have $D\geq t+1$. By our Condition 3 in the Lemma, that $(s+1)\cdot\epsilon(\lambda')\leq\frac{1}{2}$, with probability at least $\frac{1}{2}$ we have $D\geq t+1$ given a sample from $\mathcal{D}_2$, in the standard model.

By an averaging argument, it follows that with probability at least $\frac{1}{2}\cdot\frac{1}{2}=\frac{1}{4}$ over sampling T_0, the probability p_T for the event where $D\geq t+1$, is at least $\frac{1}{2}\cdot\frac{1}{2}=\frac{1}{4}$. Let us call this set of superspaces T_0, "the good set" of samples, which by definition has fraction at least $\frac{1}{4}$. Recall two facts: (1) the dimension of $(T_0)^\perp$ is t, (2) the dimension D aggregates vectors inside $S_0^\perp$. The two facts together imply that in the event $D\geq t+1$, it is necessarily the case that there exists an execution index $i\in[\ell]$ in the reduction $\mathcal{B}$ where $\mathcal{A}$ outputs a vector in $\left(S_0^\perp\setminus T_0^\perp\right)$, when given a sample from $\mathcal{D}_2$.

For every T_0 inside the good set we thus know that with probability $\frac{1}{4}$, one of the output vectors of $\mathcal{A}$ will be in $\left(S_0^\perp\setminus T_0^\perp\right)$. Since these are ℓ i.i.d. executions of $\mathcal{A}$, by union bound, for every T_0 inside the good set, when we prepare access of F_T and execute $\mathcal{A}$, we will get a vector in $\left(S_0^\perp\setminus T_0^\perp\right)$ with probability $\geq\frac{1}{4\cdot\ell}$. We deduce that for a uniformly random T_0, the probability for the output of $\mathcal{A}$ to be in $\left(S_0^\perp\setminus T_0^\perp\right)$, is at least the probability for this event (i.e., the output vector is in $\left(S_0^\perp\setminus T_0^\perp\right)$) intersecting with the event that T_0 is inside the good set, which in turn is at least

$$\frac{1}{4}\cdot\frac{1}{4\cdot\ell}=\frac{1}{16\cdot\ell}:=\frac{\epsilon}{16\cdot k\cdot(t+1)} \ ,$$

which finishes our proof. □

3 Short One-Shot Signatures Relative to a Classical Oracle

In this section we present our construction and security proof with respect to a classical oracle. We construct one-shot signatures that can sign on λ-bit messages and has quantum signing keys of size $\Theta(\lambda)$ qubits, and where the probability to forge is bounded by $2^{-\Omega(\lambda)}$, for every quantum algorithm making a polynomial number of queries q. We first describe our core construction in 17, and then our OSS scheme itself in 19, by relying on the core construction.

Construction 17. *Let $\lambda\in\mathbb{N}$ the statistical security parameter. Define $s:=18\cdot\lambda$ and let $n,r,k\in\mathbb{N}$ such that $r:=s\cdot\lambda$, $n:=r+\frac{3}{2}\cdot s$, $k:=\frac{3}{2}\cdot s+\lambda$.*

Let $\Pi:\{0,1\}^n\to\{0,1\}^n$ be a random permutation and let $F:\{0,1\}^r\to\{0,1\}^{\mathsf{poly}(r)}$ a random function. Let $H(x)$ denote the first r output bits of $\Pi(x)$, and $J(x)$ denote the remaining $n-r$

bits, which are interpreted as a vector in $\mathbb{Z}_2^{n-r}$*. For each* $y \in \{0,1\}^r$*, using the output randomness of* $F(y)$*, we sample* $\mathbf{A}(y) \in \mathbb{Z}_2^{k\times(n-r)}$ *and* $\mathbf{b}(y) \in \mathbb{Z}_2^k$*. The matrix* $\mathbf{A}(y) \in \mathbb{Z}_2^{k\times(n-r)}$ *is random with full column rank* $n-r$*, and also such that its bottom* λ *rows have full row rank* λ*.* $\mathbf{b}(y) \in \mathbb{Z}_2^k$ *is a uniformly random vector. Then, let*

$$\mathcal{P} : \{0,1\}^n \to \left(\{0,1\}^r \times \mathbb{Z}_2^k\right), \quad \mathcal{P}^{-1} : \left(\{0,1\}^r \times \mathbb{Z}_2^k\right) \to \{0,1\}^n, \quad \mathcal{D}_1 : \left(\{0,1\}^r \times \mathbb{Z}_2^k\right) \to \mathbb{Z}_2^\lambda \ ,$$

be the following oracles:

$$\mathcal{P}(x) = (\ y\ ,\ \mathbf{A}(y)\cdot J(x) + \mathbf{b}(y)\ ) \ \text{ where } y = H(x)$$

$$\mathcal{P}^{-1}(y,\mathbf{u}) = \begin{cases} \Pi^{-1}(y,\mathbf{z}) & \exists\, \mathbf{z} \in \mathbb{Z}_2^{n-r} : \mathbf{A}(y)\cdot\mathbf{z} + \mathbf{b}(y) = \mathbf{u} \\ \bot & \text{else} \end{cases}$$

$$\mathcal{D}(y,\mathbf{v}) = \begin{cases} \mathbf{c}_{y,\mathbf{v}} & \text{if } \mathbf{v}^T\cdot\mathbf{A}(y) \in \mathsf{RowSpan}\left(\mathbf{A}(y)_{[(k-\lambda)+1]},\cdots,\mathbf{A}(y)_{[(k-\lambda)+\lambda]}\right) \\ \bot & \text{otherwise} \end{cases}$$

where for $\mathbf{A} \in \mathbb{Z}_2^{k\times(n-r)}$ *and* $j \in [k]$*,* $\mathbf{A}_{[j]} \in \mathbb{Z}_2^{n-r}$ *is the* j*-th row of* $\mathbf{A}$ *(e.g.,* k*-th row is bottom), and* $\mathbf{c}_{y,\mathbf{v}} \in \mathbb{Z}_2^\lambda$ *is the coordinates vector, of the vector* $\mathbf{v}^T\cdot\mathbf{A}(y) \in \mathbb{Z}_2^{n-r}$ *with respect to the basis* $\mathbf{A}(y)_{[(k-\lambda)+1]},\cdots,\mathbf{A}(y)_{[(k-\lambda)+\lambda]}$ *(i.e., element* $j \in [\lambda]$ *of* $\mathbf{c}_{y,\mathbf{v}}$ *is the coefficient of* $\mathbf{A}(y)_{[(k-\lambda)+j]}$*).*

We next describe our one-shot signature scheme explicitly. As part of our scheme we will use the following property of error-correcting codes:

Theorem 18 ([Pin65])**.** *There exists a universal constant* $C > 0$ *such that, for any constant* $\epsilon > 0$*, a random linear binary code with rate* $C\cdot\epsilon^2$ *has, except with probability negligible in the length of the code, a minimum distance at least* $1/2 - \epsilon$*. For a constant* $\epsilon > 0$ *we denote by* Encode_ϵ *the ECC's encoding algorithm, which is efficient and just samples a random linear code with rate* $C\cdot\epsilon^2$ *and encodes any message* $m \in \{0,1\}^\lambda$ *to* $\mathsf{Encode}_\epsilon(m) \in \{0,1\}^{\frac{\lambda}{C\cdot\epsilon^2}}$*.*

Construction 19 (One-Shot Signature Construction Relative to a Classical Oracle)**.** *Our one-shot signature (OSS) scheme* $(\mathsf{Setup},\mathsf{TokenGen},\mathsf{Sign},\mathsf{Ver})$*, is defined for security parameter* $\lambda \in \mathbb{N}$ *and can sign on messages in* $\{0,1\}^{\lambda'}$*, for* $\lambda' := C\cdot\epsilon^2\cdot\lambda$ *such that* $C > 0$ *is the constant from Theorem 18 and for* $\epsilon := \frac{1}{3}$*. Our scheme is defined as follows.*

- $\left(\mathcal{P},\mathcal{P}^{-1},\mathcal{D}\right) \leftarrow \mathsf{Setup}\left(1^\lambda\right)$*: The oracle that the setup algorithm samples is exactly the oracles sampled by Construction 17.*
- $(\mathsf{pk},|\mathsf{sk}\rangle) \leftarrow \mathsf{TokenGen}^{\mathcal{P},\mathcal{P}^{-1}}$*: To sample a signature token, start with a uniform superposition* $|+\rangle^{\otimes n}$*, apply* $\mathcal{P}$ *to obtain*

$$\frac{1}{2^{\frac{n}{2}}}\sum_{x\in\{0,1\}^n} |x, H(x), \mathbf{u}_x\rangle \ ,$$

where $H(\cdot)$ *is the hash function from Construction 17 and for* $x \in \{0,1\}^n$*, the vector* $\mathbf{u}_x \in \mathbb{Z}_2^k$ *is the output vector of the oracle* $\mathcal{P}$*.*

Next, use $\mathcal{P}^{-1}$ *to un-compute the register holding* x *and then measure the register holding* $H(x) \in \{0,1\}^r$*, to obtain the state*

$$|y\rangle \otimes \frac{1}{2^{\frac{n-r}{2}}} \sum_{\mathbf{u} \in (\mathsf{ColSpan}(\mathbf{A}(y)) + \mathbf{b}(y))} |\mathbf{u}\rangle \ .$$

Finally, set $\mathsf{pk} := y$ *and* $|\mathsf{sk}\rangle := \frac{1}{2^{\frac{n-r}{2}}} \sum_{\mathbf{u} \in (\mathsf{ColSpan}(\mathbf{A}(y)) + \mathbf{b}(y))} |\mathbf{u}\rangle$*.*

- $\sigma \leftarrow \mathsf{Sign}^{\mathcal{P}^{-1},\mathcal{D}}\left(\mathsf{pk}, |\mathsf{sk}\rangle, m \in \{0,1\}^{\lambda'}\right)$*: We describe our signing algorithm in 20.*
- $\mathsf{Ver}^{\mathcal{P}^{-1}}\left(\mathsf{pk}, m \in \{0,1\}^{\lambda'}, \sigma\right) \in \{0,1\}$*: Parse* pk *as* $y \in \{0,1\}^r$ *and parse* σ *as* $\mathbf{u}_\sigma \in \mathbb{Z}_2^k$*. Encode* $m \in \{0,1\}^{\lambda'}$ *into* $\mathsf{Encode}_\epsilon(m) := m^* \in \{0,1\}^\lambda$ *using the ECC from Theorem 18. Output* 1 *iff both,* $\mathcal{P}^{-1}(y, \mathbf{u}_\sigma) \neq \perp$ *and* m^* *has Hamming distance bounded by* $\lambda \frac{\left(\frac{1}{2}-\epsilon\right)}{2} := \frac{\lambda}{12}$ *to the last* λ *bits of* σ*.*

Signing Multi-bit Messages. We next describe a quantum signing algorithm that can sign $\Omega(\lambda)$-bit messages, and furthermore does so in "parallel". Specifically, the algorithm works in iterations, and in each iteration, the algorithm makes two queries to $\mathcal{D}$ and elsewhere executes in constant parallel time. In particular, it is parallel in the sense that signing for bit $j+1$ does not need to wait for the signing of bit j. Our algorithm takes constant iterations and produces a valid signature with probability exponentially close to 1 (in the security parameter).

The intuition behind the algorithm originates in the "measure and correct" signing algorithm of [Shm22b], which in turn is inspired by the OSS signing procedure of [AGKZ20]. Our algorithm first encodes m to a large message $m^* \in \{0,1\}^\lambda$ using an ECC. Then, the algorithm tries to sign by collapsing all λ qubits to the wanted value m^*. Each qubit falls to its right value with probability $1/2$, independently of the other qubits, so we expect roughly half the qubits to collapse to the right value when we do this. Then, the signing algorithm "corrects" by returning the corresponding qubits (i.e., the ones that need correction) to full superposition using the dual oracle. This correction will let us retry to collapse the qubits that did not collapse correctly for the first time, to their correct value with accordance to m. After some tries, we expect some minimal fraction of qubits to fall to their correct classical value. The formal procedure follows.

Construction 20 (A many-bit parallel quantum signing algorithm for coset states)**.** *We describe the signing algorithm* $\mathsf{Sign}^{\mathcal{P}^{-1},\mathcal{D}}\left(\mathsf{pk}, |\mathsf{sk}\rangle, m \in \{0,1\}^{\lambda'}\right)$ *of the OSS scheme.*

1. *Denote by* $\mathcal{U}$ *the* k*-qubit register holding* $|\mathsf{sk}\rangle$*. Parse* pk *as* $y \in \{0,1\}^r$ *and continue only if* $\mathcal{P}^{-1}(y, \mathcal{U}) \neq \perp$*.*
2. *Encode the message* m *into the codeword* $\mathsf{Encode}_\epsilon(m) \to m^* \in \{0,1\}^\lambda$ *using the ECC.*
3. *For* t *going from* 1 *until* 4*:*
   (a) *Make a measurement on the last* λ *qubits of* $\mathcal{U}$*, denote it by* $m^t \in \{0,1\}^\lambda$*.*
   (b) *Execute* $H^{\otimes k}$ *on the register* $\mathcal{U}$*.*

(c) *Initialize a λ-qubit register $\mathcal{C}$ with zeros. Apply $\mathcal{D}(y,\cdot)$ to $\mathcal{U}$, putting the output in $\mathcal{C}$. Measure only the qubits $\mathcal{C}_i$ such that $m^*_i \oplus m^t_i = 1$. Discard the measurement results and un-compute the information that's inside the register $\mathcal{C}$ by using the oracle $\mathcal{D}(y,\cdot)$ again.*

(d) *Execute $H^{\otimes k}$ again on the register $\mathcal{U}$. Increment the loop variable t by 1.*

4. *measure the rest of the register $\mathcal{U}$ and let $\sigma \in \{0,1\}^k$ the measurement result.*

Claim 21 (Correctness of Signing Algorithm)**.** *With probability $1-2^{-\Omega(\lambda)}$ over sampling the linear code in the ECC scheme, we have that the probability for the event of getting a valid signature over the random experiment of (1) sampling $(\mathsf{pk}, |\mathsf{sk}\rangle) \leftarrow \mathsf{TokenGen}^{\mathcal{P},\mathcal{P}^{-1}}$ and then (2) executing the signing algorithm, is $1-2^{-\Omega(\lambda)}$.*

Proof. The correctness proof has two steps. The first and main part, proves that in every iteration of the algorithm, for the qubits of $\mathcal{U}$ that did not collapse to the correct value of m^*, the algorithm will return these to uniform superposition at the end of that same iteration. In the second part, we explain how a simple concentration bound proves that we get a valid signature with high probability, by using the first part of correctness.

Correction of faulty collapsed bits back to superposition. We will show that the algorithm can make repeated tries for the target bits that were not signed correctly, by "re-entropization" of the qubits that collapsed to the wrong value, without re-entropizing (or changing the value) of the bits that collapsed to the correct value. The correctness of the algorithm is conditioned on the sampled matrix $\mathbf{A}(y) \in \mathbb{Z}_2^{k\times(n-r)}$ having full rank (i.e., column-rank $n-r$), its bottom λ rows having full rank as well (i.e. row-rank λ), and on the ECC having minimal distance of $\lambda \cdot \left(\frac{1}{2}-\epsilon\right) := \frac{\lambda}{6}$. The probability that any of these do not happen is $2^{-\Omega(\lambda)}$.

Next, for $y \in \{0,1\}^r$,

- Let $S_{y,0} := \mathsf{ColSpan}(\mathbf{A}(y))$, which has dimension $n-r$.
- Let S_y the subspace of $S_{y,0}$ with vectors such that their last λ bits are all 0. Accordingly, S_y has dimension $n-r-\lambda$.
- For $j \in [\lambda]$, let $S_{y,j}$ the subspace of $S_{y,0}$ with vectors such that their last λ bits are all 0, except bit $(k-\lambda)+j$, which may be arbitrary. Note that since the bottom λ rows of $\mathbf{A}(y)$ are full rank, it follows that $S_{y,j}$ has dimension $n-r-\lambda+1$ and thus satisfies $S_y \subsetneq S_{y,j} \subsetneq S_{y,0}$.

Formally, considering our signing algorithm from 20, for every iteration t, observe that at the end of Step 3a, the state we have in register $\mathcal{U}$ is of the form

$$\sum_{\mathbf{u}\in S_y} (-1)^{\langle z^t,\mathbf{u}\rangle} \cdot |x^t + \mathbf{b}(y) + \mathbf{u}\rangle \tag{1}$$

for some $z^t \in S_y^\perp$ and $x^t \in S_{y,0}$. As an example, after the first iteration, we get the above state for $z^1 = 0^k$ and $x^1 \in S^{y,0}$ such that for the measurement result $m^1 \in \{0,1\}^\lambda$ and for every $j \in [\lambda]$ we have $[x^1 \oplus \mathbf{b}(y)]_{(k-\lambda)+j} = m^1_j$.

We will show that at the end of the iteration t (after Step 3d), after re-entropizing the bits $j \in [\lambda]$ such that $m^*_j \oplus m^t_j = 1$, the state in $\mathcal{U}$ will be of the form

$$\sum_{\mathbf{u}\in\mathcal{S}_{(m^*,\,m^t)}} (-1)^{\langle z^{t+1},\mathbf{u}\rangle} \cdot |x^t + \mathbf{b}(y) + \mathbf{u}\rangle \ . \tag{2}$$

for some $z^{t+1} \in S_y^{\perp}$, and where $\mathcal{S}_{(m^*,\, m^t)} := \mathsf{Span}\left(\{S_{y,j}\}_{j\,:\,[m^* \oplus m^t]_j = 1}\right)$. It can be verified by the reader that if we can correct to the above state, this lets us proceed for attempting to signing the rest of the bits of m correctly, while keeping the already-correctly-collapsed bits.

So, assume we have a state as per Equation 1, which happens at iteration t after Step 3a. Next, after executing Step 3b our state is

$$\sum_{\mathbf{v} \in S_y^{\perp}} (-1)^{\langle x^t + \mathbf{b}(y), \mathbf{v} \rangle} \cdot |z^t + \mathbf{v}\rangle \ ,$$

and after the next Step 3c, the state makes a partial collapse to the form

$$\sum_{\mathbf{v} \in \mathcal{S}^{\perp}_{(m^*,\, m^t)}} (-1)^{\langle x^t + \mathbf{b}(y), \mathbf{v} \rangle} \cdot |z^{t+1} + \mathbf{v}\rangle \ ,$$

such that $z^{t+1} \in S_y^{\perp}$.

Specifically (and for the interested reader), if we look at the $|m^* \oplus m^t|$ measurement results (which we discarded) from Step 3c, z^{t+1} is such that for the vector $\left(z^{t+1}\right)^T \cdot \mathbf{A}(y) \in \mathbb{Z}_2^{n-r}$, its coordinates vector $\mathbf{c}_{y,z^{t+1}} \in \mathbb{Z}_2^{\lambda}$ with respect to the basis $\mathbf{A}(y)_{[(k-\lambda)+1]}, \cdots, \mathbf{A}(y)_{[(k-\lambda)+\lambda]}$ has the same values as what was measured in Step 3c. Finally, since after Step 3d the state in register $\mathcal{U}$ is indeed of the form described by Equation 2, this finishes the first part of our proof.

Probability analysis of getting a valid signature. The remainder of the proof simply uses Chernoff's concentration bound on the event that a bit in uniform superposition collapsed to its right value w.r.t. m^*. Specifically, we proves that at the beginning of every iteration t, for each of the last λ qubits of $\mathcal{U}$, if is either in uniform superposition (and independently of all of the other λ last qubits of $\mathcal{U}$), or it is classical and have the value of the corresponding bit of m^*. So, since it is in particular in uniform superposition, when measured its probability for collapsing to the correct value of m^* is exactly $1/2$. Accordingly, due to the fact that these are statistically independent events (thanks to the fact that the row rank of the last λ rows of $\mathbf{A}(y)$ is full and equals to λ), with probability $1 - 2^{-\Omega(\lambda)}$, roughly half of the qubits that were in superposition collapse to their correct value of m^*.

This means that after 4 iterations of the algorithm, with probability $1 - 2^{-\Omega(\lambda)}$, Chernoff's bound implies that at most a $\frac{1}{16} + \frac{1}{100}$ fraction of the bits of m^t (for $t = 4$) do not match the corresponding bits of m^*. It follows in particular that with probability $1 - 2^{-\Omega(\lambda)}$, after 4 iterations the resulting signature $\sigma \in \{0,1\}^k$ has less than $\frac{\lambda}{12}$ Hadamard distance to m^*, which means that the signature will be verified successfully by the signature verification algorithm. □

Security, and comparison to the security reduction from [SZ25]. In order to prove security (that is, strong unforgeability of the OSS), it will be enough to prove the collision resistance of the function H from the oracle $\mathcal{P}$. The structure of the proof and high-level strategy remains the same as in [SZ25]: In the first part we show that a collision finder given $\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}\right)$ can be transformed into a collision finder against the dual-free setting $\left(\mathcal{P}, \mathcal{P}^{-1}\right)$, this is proved in our Theorem 25. The second part proves that collision finding in $\left(\mathcal{P}, \mathcal{P}^{-1}\right)$ is hard, by reducing to the collision-resistance of other functions. There are two points of meaningful difference from the previous proof – one change to each of the above conceptual parts of the proof.

The first change to the proof (compared to [SZ25]) is due to the adversary getting more information from the oracle $\mathcal{D}$: λ bits instead of 1. We need to prove that also here, the function H (computed inside $\mathcal{P}$) is collision resistant. The new proof handles this by executing the same high-level ideas from the reduction from previous work, but slightly more carefully, and by adding a new generalized variation of subspace-hiding (given formally in Section 2.3), which we call subspace-hiding functions. After completing the first part of the proof, we got rid of the dual $\mathcal{D}$ and all that remains is to prove that the dual-free oracles $\left(\mathcal{P}, \mathcal{P}^{-1}\right)$ are collision-resistant.

The second change is that the dimension k, which is the dimension of the space that the random cosets $\mathsf{ColSpan}\left(\mathbf{A}(y)\right) + \mathbf{b}(y)$ live in, is now asymptotically smaller (and even optimal). Formally, k is now on the order of $\Theta\left(\lambda\right)$, where in previous work [SZ25] it was on the order of $\Theta\left(\lambda^2\right)$. Here, things are conceptually different: the previous technique for proving the collision resistance of $\left(\mathcal{P}, \mathcal{P}^{-1}\right)$ relies solely on a parallel repetition of random 2-to-1 functions, and it breaks down in this setting (see an elaborated intuitive explanation in Section 1.2). Nonetheless, in Section 3.3 we show a new reduction and technique that resolves this.

Overall, we obtain the following main security Theorem.

Theorem 22 (Collision Resistance of H)**.** *Let $\mathcal{O}_{n,r,k}$ the distribution over oracles defined in Construction 17. Let $\mathcal{A}$ an oracle aided q-query (computationally unbounded) quantum algorithm. Then,*

$$\Pr\left[x_0 \neq x_1 \wedge H(x_0) = H(x_1) \ : \ \begin{matrix}\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}\right) & \leftarrow \mathcal{O}_{n,r,k} \\ (x_0, x_1) & \leftarrow \mathcal{A}^{\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}}\end{matrix}\right] \leq O\left(\frac{\lambda \cdot k^2 \cdot q^3}{2^\lambda}\right) \ .$$

Proof. Assume towards contradiction that there is an oracle aided quantum algorithm $\mathcal{A}$, making q queries, that given a sample oracle $\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}\right) \leftarrow \mathcal{O}_{n,r,k}$ outputs a collision (x_0, x_1) in H with probability ϵ, such that $\epsilon \geq \omega\left(\frac{\lambda \cdot k^2 \cdot q^3}{2^\lambda}\right)$.

Note that by our parameter choices in Construction 17 and by our assumption towards contradiction $\epsilon \geq \omega\left(\frac{\lambda \cdot k^2 \cdot q^3}{2^\lambda}\right)$, one can verify through calculation that for $s' := s - (n - r - s)$ we have (1) $\frac{\lambda \cdot k^8 \cdot q^8 \cdot s}{\sqrt{2^{n-r-\lambda-s}}} \leq o\left(\epsilon^4\right)$, and also (2) $\frac{k^2 \cdot q^3 \cdot (n-r-s)}{2^{s'}} \leq o\left(\epsilon^2\right)$. This means that the conditions of Theorem 25 are satisfied, and it follows there is a q-query algorithm $\mathcal{B}$ that gets access only to $\left(\mathcal{P}, \mathcal{P}^{-1}\right)$, sampled from $\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}\right) \leftarrow \mathcal{O}_{r+s,\, r,\, k-(n-r-s)}$ that finds collisions in H with probability $\geq \frac{\epsilon}{2^6 \cdot k^2}$.

Now, consider the statement of Theorem 28, with the following interface: "n" in the Theorem will be $r + s$ here, "r" in the theorem stays the same and is r here, and "k" in the Theorem is $k - (n - r - s)$ here. Note that our parameter choices in Construction 17 imply that $\frac{r+s}{(r+s)-r}$ and thus "$\frac{n}{n-r}$" from the theorem is an integer, and also $k - (n - r - s) \geq (r + s) - r + \lambda$ and thus "$k \geq n - r + \lambda$" from the theorem. It follows by Theorem 28 that there is a q-query algorithm $\mathcal{B}'$ that given oracle access to random claw-free permutation $H^* : \{0,1\}^{\lambda+1} \to \{0,1\}^\lambda$ (as in Definition 27), finds a collision in H^* with probability $\frac{\epsilon}{2^6 \cdot k^2 \cdot (n-r)}$.

However, we know that finding collisions in H^* is hard: It follows by Lemma 29 that

$$\frac{\epsilon}{2^6 \cdot k^2 \cdot (n-r)} \leq O\left(\frac{q^3}{2^{\frac{r}{n-r}}}\right) \ ,$$

which in turn implies

$$\epsilon \leq O\left(\frac{\lambda \cdot k^2 \cdot q^3}{2^\lambda}\right) \ ,$$

in contradiction to $\epsilon \geq \omega\left(\frac{\lambda \cdot k^2 \cdot q^3}{2^\lambda}\right)$. □

3.1 Bloating the Dual

Let $\mathcal{O}'_{n,r,k,s}$ denote the following distribution over $\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}'$. The oracles $\mathcal{P}, \mathcal{P}^{-1}$ are defined identically to $\mathcal{O}_{n,r,k}$, and the oracle $\mathcal{D}$ will change. Now, for $s \leq n - r - \lambda$, we let $\mathbf{A}(y)^{(0)} \in \mathbb{Z}_2^{k \times s}$ denote the first (i.e. leftmost) s columns of $\mathbf{A}(y) \in \mathbb{Z}_2^{k \times (n-r)}$ and $\mathbf{A}(y)^{(1)} \in \mathbb{Z}_2^{k \times (n-r-s)}$ denote the remaining $n - r - s$ (rightmost) columns. In the bloated dual, we do the same things as in the original $\mathcal{D}$, but with respect to $\mathbf{A}(y)^{(1)} \in \mathbb{Z}_2^{k \times (n-r-s)}$ rather than with respect to $\mathbf{A}(y) \in \mathbb{Z}_2^{k \times (n-r)}$. Formally:

$$\mathcal{D}'(y, \mathbf{v}) = \begin{cases} \mathbf{c}_{y,\mathbf{v}} & \text{if } \mathbf{v}^T \cdot \mathbf{A}(y)^{(1)} \in \mathsf{RowSpan}\left(\mathbf{A}(y)^{(1)}_{[(k-\lambda)+1]}, \cdots, \mathbf{A}(y)^{(1)}_{[(k-\lambda)+\lambda]}\right) \\ \perp & \text{otherwise} \end{cases}$$

where here, similarly to the construction (but not quite the same), $\mathbf{c}_{y,\mathbf{v}} \in \mathbb{Z}_2^\lambda$ is the coordinates vector, of the vector $\mathbf{v}^T \cdot \mathbf{A}(y)^{(1)} \in \mathbb{Z}_2^{n-r-s}$ with respect to the basis $\mathbf{A}(y)^{(1)}_{[(k-\lambda)+1]}, \cdots, \mathbf{A}(y)^{(1)}_{[(k-\lambda)+\lambda]}$.

Lemma 23. *Suppose there is an oracle aided q-query quantum algorithm $\mathcal{A}$ such that*

$$\Pr\left[(y_0 = y_1) \wedge (x_0 \neq x_1) \ : \ \begin{array}{rl} \left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}\right) & \leftarrow \mathcal{O}_{n,r,k} \\ (x_0, x_1) & \leftarrow \mathcal{A}^{\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}} \\ (y_b, \mathbf{u}_b) & \leftarrow \mathcal{P}(x_b) \end{array}\right] \geq \epsilon \ .$$

Also, let $s \in \mathbb{N}$ such that $s \leq n - r - \lambda$, define $s' := s - (n - r - s)$, and assume

1. $\frac{\lambda \cdot k^8 \cdot q^8 \cdot s}{\sqrt{2^{n-r-s-\lambda}}} \leq o\left(\epsilon^4\right)$,
2. $\frac{k^2 \cdot q^3 \cdot (n-r-s)}{2^{s'}} \leq o\left(\epsilon^2\right)$.

Then,

$$\Pr\left[\begin{array}{l} (y_0 = y_1 := y) \wedge \\ (\mathbf{u}_0 - \mathbf{u}_1) \notin \mathsf{ColSpan}\left(\mathbf{A}(y)^{(1)}\right) \end{array} \ : \ \begin{array}{rl} \left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}'\right) & \leftarrow \mathcal{O}'_{n,r,k,s} \\ (x_0, x_1) & \leftarrow \mathcal{A}^{\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}'} \\ (y_b, \mathbf{u}_b) & \leftarrow \mathcal{P}(x_b) \end{array}\right] \geq \frac{\epsilon}{2^6 \cdot k^2} \ .$$

Proof. Assume there is an oracle-aided q-query quantum algorithm $\mathcal{A}$ that given oracle access to $\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}\right) \leftarrow \mathcal{O}_{n,r,k}$ outputs a pair (x_0, x_1) of n-bit strings. Denote by ϵ the probability that x_0, x_1 are both distinct and collide in $H(\cdot)$ (i.e., their y-values are identical). We next define a sequence of hybrid experiments, outputs and success probabilities for them, and explain why the success probability in each consecutive pair is statistically close.

- Hyb_0: The original execution of $\mathcal{A}$.

The process Hyb_0 is the above execution of $\mathcal{A}$ on input oracles $\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}\right)$. We define the output of the process as (x_0, x_1) and the process execution is considered as successful if x_0, x_1 are both distinct and collide in $H(\cdot)$. By definition, the success probability of Hyb_0 is ϵ.

- Hyb_1: Simulating the oracles using only a bounded number of cosets $(\mathbf{A}(y), \mathbf{b}(y))$, by using small-range distribution.

Consider the function F which samples for every $y \in \mathbb{Z}_2^r$ the i.i.d. coset description $(\mathbf{A}(y), \mathbf{b}(y))$. These cosets are then used in all three oracles $\mathcal{P}$, $\mathcal{P}^{-1}$ and $\mathcal{D}$. The difference between the current hybrid and the previous hybrid is that we swap F with F' which is sampled as follows: We set $R := \left(300 \cdot q^3\right) \cdot \frac{2^7 \cdot k^2}{\epsilon}$ and for every $y \in \mathbb{Z}_2^r$ we sample a uniformly random $i_y \leftarrow [R]$, then sample for every $i \in [R]$ a coset $\left(\mathbf{A}(i) \in \mathbb{Z}_2^{k \times (n-r)}, \mathbf{b}(i) \in \mathbb{Z}_2^k\right)$ as usual. For $y \in \mathbb{Z}_2^r$ we define $F'(y) := (\mathbf{A}(i_y), \mathbf{b}(i_y))$ instead of the previous $F(y) := (\mathbf{A}(y), \mathbf{b}(y))$.

By the small-range distribution technique [Zha21a] (and concretely, Theorem A.6 from [AGQY22]), it follows that for every quantum algorithm making at most q queries and tries to distinguish between F and F', the distinguishing advantage is bounded by $\frac{300 \cdot q^3}{R} < \frac{\epsilon}{8}$, which means in particular that the outputs of this hybrid and the previous one has statistical distance bounded by $\frac{\epsilon}{8}$. It follows that the success probability of the current hybrid is $:= \epsilon_1 \geq \epsilon - \frac{\epsilon}{8} = \frac{7 \cdot \epsilon}{8}$.

- Hyb_2: Relaxing dual verification oracle to accept a larger subspace, by subspace hiding functions.

The change we make in this hybrid is that we make the membership check dual oracles more relaxed. Formally, we invoke Lemma 14 with the following interface, and for every $i \in [R]$. The smallest oracle "S_0" from the Lemma 14 statement is simply $S_{i,0}^{\perp} := \mathsf{ColSpan}\left(\mathbf{A}(i)\right)^{\perp}$ here and "S" from the lemma is $S_i^{\perp}$, defined as the total set of vectors that $\mathcal{D}$ does not output $\perp$ on. The partition "P" from the lemma statement is the partition defined by the output of the oracle $\mathcal{D}$ (observe that this is indeed a partition into cosets of S that are all parallel to S_0). That is, the oracle $\mathcal{D}$ computes the function F_S for these subspaces.

Now, for every $i \in [R]$, after sampling $(\mathbf{A}(i), \mathbf{b}(i))$, we sample superspace $T_{i,0}^{\perp}$ and by the statement of Lemma 14, (1) $T_{i,0}^{\perp}$ is a random superspace of $S_{i,0}^{\perp}$ (which itself has dimension $k-(n-r)$) with $k-(n-r-s)$ dimensions, and (2) $T_i^{\perp}$ is the joint span of $T_{i,0}^{\perp}$ and $S_i^{\perp}$. For concreteness and because it will later be useful to our proof, assume that our process of sampling $T_{i,0}^{\perp}$ is by sampling its dual $T_{i,0}$, which in turn is sampled by sampling a uniformly random invertible $\mathbf{M}_i \in \mathbb{Z}_2^{(n-r)\times(n-r)}$, multiplying by $\mathbf{A}(i)$ and then taking the rightmost $n-r-s$ columns of the generated matrix.

By the guarantees of Lemma 14, for every $i \in [R]$, changing $\mathcal{D}$ to compute F_T is $\left(\frac{\lambda \cdot q \cdot s}{\sqrt{2^{n-r-\lambda-s}}}\right)$-indistinguishable, for any q-query algorithm. Since we use the above indistinguishability R times, we get $R \cdot \left(\frac{\lambda \cdot q \cdot s}{\sqrt{2^{n-r-\lambda-s}}}\right)$-indistinguishability. It follows that the success probability of the current hybrid is $:= \epsilon_2 \geq \epsilon_1 - R \cdot \left(\frac{\lambda \cdot q \cdot s}{\sqrt{2^{n-r-\lambda-s}}}\right) \geq \frac{7 \cdot \epsilon}{8} - O\left(\frac{\lambda \cdot k^2 \cdot q^4 \cdot s \cdot \frac{1}{\epsilon}}{\sqrt{2^{n-r-\lambda-s}}}\right)$, which in turn by Condition 1 in our Lemma's statement, is at least $\frac{3 \cdot \epsilon}{4}$.

- Hyb_3: For every $i \in [R]$, asking for the sum of collisions to be outside of $T_{i,0}$, by using dual-subspace anti-concentration.

In the current hybrid we change the success predicate of the experiment. Recall that as part of sampling the oracles in the previous hybrid, we sample R i.i.d. cosets $(\mathbf{A}(i), \mathbf{b}(i))_{i \in [R]}$ which are used in all three oracles $\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}\right)$. We then sample R i.i.d. $(k-n+r+s)$-dimensional superspaces

$\left(T_{i,0}^{\perp}\right)_{i\in[R]}$ of the R corresponding duals $\left(S_{i,0}^{\perp}\right)_{i\in[R]}$, where for every i, $S_{i,0} := \mathsf{ColSpan}\left(\mathbf{A}(i)\right)$. The change we make to the success predicate in the current hybrid is the following: at the end of the execution we get a pair (x_0, x_1) from $\mathcal{A}$. We define the process as successful if $y_0 = y_1 := y$ and also $(\mathbf{u}_0 - \mathbf{u}_1) \notin T_{i_y,0}$, rather than only asking that $x_0 \neq x_1$.

Note that $\mathcal{A}$ finds collisions with probability ϵ_2 in the previous hybrid Hyb_2 (and since this hybrid is no different, the same goes for the current hybrid), which means it finds collisions (x_0, x_1) such that $y_0 = y_1 := y$ and $(\mathbf{u}_0 - \mathbf{u}_1) \in S_{i_y,0}$. For every value $i \in [R]$ denote by $\epsilon_2^{(i)}$ the probability to find a collision in index i, or formally, to find $x_0 \neq x_1$ such that $y_0 = y_1 := y$, $i_y = i$ and $(\mathbf{u}_0 - \mathbf{u}_1) \in S_{i_y,0}$. We deduce $\sum_{i\in[R]} \epsilon_2^{(i)} = \epsilon_2$. Let L be a subset of indices $i \in [R]$ such that $\epsilon_2^{(i)} \geq \frac{\epsilon_2}{2\cdot R}$ and note that $\sum_{i\in L} \epsilon_2^{(i)} \geq \frac{\epsilon_2}{2}$. Let ϵ_3 be the success probability of the current hybrid. For every value $i \in [R]$ also denote by $\epsilon_3^{(i)}$ the probability to find a collision such that $y_0 = y_1 := y$, $(\mathbf{u}_0 - \mathbf{u}_1) \notin T_{i_y,0}$ and also $i_y = i$. We deduce $\sum_{i\in[R]} \epsilon_3^{(i)} = \epsilon_3$.

We would now like to use Lemma 16, specifically on $S_{i,0}^{\perp}$ as the static subspace and $T_{i,0}^{\perp}$ as its random superspace, and use the lemma for the output of the adversary to avoid the dual $T_{i,0}$ of the superspace $T_{i,0}^{\perp}$. So, we make sure that we satisfy its requirements. Let any $i \in L$, we know that by definition $\epsilon_2^{(i)} \geq \frac{\epsilon_2}{2\cdot R}$ and also recall that $\epsilon_2 \geq \frac{3\cdot\epsilon}{4}$, $R := \left(300 \cdot q^3\right) \cdot \frac{2^7\cdot k^2}{\epsilon}$ and thus

$$\epsilon_2^{(i)} \geq \frac{\epsilon_2}{2\cdot R} \geq \frac{3\cdot\epsilon}{8} \cdot \frac{1}{R} \geq \Omega\left(\frac{\epsilon^2}{q^3\cdot k^2}\right) \quad .$$

Note the following parameter interfaces between our setting and the setting of the Lemma: "k", "s" and "λ" in the lemma are the same k, s and λ here and "r" in the lemma is $k-(n-r)$ here. Let $s' := s - (n - r - s)$ and for any $i \in L$ let $\ell_i := \frac{k^2}{\epsilon_2^{(i)}} \leq O\left(\frac{k^4\cdot q^3}{\epsilon^2}\right)$. Note that by our Lemma 23 statement's conditions, by Condition 2 we have (1) $\frac{n-r-s}{2^{s'}} \leq \frac{\epsilon_2^{(i)}}{2}$, and by Condition 1 we have (2) $\frac{\lambda\cdot q\cdot \ell_i^2\cdot s}{\sqrt{2^{k-(k-(n-r))-\lambda-s}}} \leq \frac{1}{2}$. Since this satisfies Lemma 16, it follows that for every $i \in L$ we have $\epsilon_3^{(i)} \geq \frac{\epsilon_2^{(i)}}{16\cdot k^2}$. It follows that

$$\epsilon_3 = \sum_{i\in[R]} \epsilon_3^{(i)} \geq \sum_{i\in L} \epsilon_3^{(i)} \geq \sum_{i\in L} \frac{\epsilon_2^{(i)}}{16\cdot k^2} \geq \frac{\left(\frac{\epsilon_2}{2}\right)}{16\cdot k^2} \geq \frac{3\cdot\epsilon}{2^7\cdot k^2} \quad .$$

- Hyb_4: For every $i \in [R]$, de-randomizing $T_{i,0}$ and defining it as the column span of the $n-r-s$ rightmost columns of the matrix $\mathbf{A}(i)$, by using the random permutation Π and random function F.

Recall that at the basis of all oracles there are the R cosets $(\mathbf{A}(i), \mathbf{b}(i))_{i\in[R]}$. We use $\mathbf{A}(i), \mathbf{b}(i)$ as is in the oracles $\mathcal{P}$, $\mathcal{P}^{-1}$, but use $\mathbf{A}(i) \cdot \mathbf{M}_i$ in $\mathcal{D}$. We will resolve this discrepancy in this hybrid. This hybrid is the same as the previous, with one change: For every $i \in [R]$, after sampling the coset $(\mathbf{A}(i), \mathbf{b}(i))$, instead of continuing to randomly sample $T_{i,0}^{\perp}$, we simply define $T_{i,0} := \mathsf{ColSpan}\left(\mathbf{A}(i)^{(1)}\right)$ such that $\mathbf{A}(i)^{(1)} \in \mathbb{Z}_2^{k\times(n-r-s)}$ is defined to be the last $n-r-s$ columns of the matrix $\mathbf{A}(i) \in \mathbb{Z}_2^{k\times(n-r)}$. We will define intermediate hybrids $\mathsf{Hyb}_{3.1}, \mathsf{Hyb}_{3.2}$ and then explain why the previous hybrid is equivalent to $\mathsf{Hyb}_{3.1} \equiv \mathsf{Hyb}_{3.2} \equiv \mathsf{Hyb}_4$.

- **Using the random permutation Π.** For every $i \in [R]$ consider the superspace $T_{i,0}^{\perp}$, which has $k-n+r+s$ dimensions. Recall the (invertible) matrix $\mathbf{M}_i \in \mathbb{Z}_2^{(n-r)\times(n-r)}$ from previous hybrid such that $T_{i,0}$ is the columns span of the $n-r-s$ rightmost columns of $\mathbf{T}_i := \mathbf{A}(i)\cdot\mathbf{M}_i$.

  In $\mathsf{Hyb}_{3.1}$, we define the permutation Γ over $\{0,1\}^n$ defined as follows: For an input $x \in \{0,1\}^n$, it takes the left r bits denoted $y \in \mathbb{Z}_2^r$, computes $i_y \in [R]$, then applies matrix multiplication by $\mathbf{M}_{i_y}$ to the remaining right $n-r$ bits. Observe that since $\mathbf{M}_i$ is invertible for all i, then Γ is indeed a permutation. The change we make from Hyb_3 to $\mathsf{Hyb}_{3.1}$ is that in the current hybrid we apply Γ to the *output* of Π inside the execution of a query to $\mathcal{P}$, and apply Γ^{-1} to the *input* of Π^{-1} inside the execution of a query to $\mathcal{P}^{-1}$. Note that for a truly random n-bit permutation Π, concatenating any fixed permutation Γ like this is statistically equivalent to just computing Π and Π^{-1}, thus the outputs (and in particular success probabilities) between Hyb_3 and $\mathsf{Hyb}_{3.1}$ are identical.

- **Using the random function F.** In this intermediate $\mathsf{Hyb}_{3.2}$, instead of sampling $\mathbf{A}_i$ and then using it as is, we will sample it multiplied by the inverse of $\mathbf{M}_i$, that is, our sampler gives us $\mathbf{A}_i \cdot \mathbf{M}_i^{-1}$ instead of $\mathbf{A}_i$. This is statistically equivalent due to the randomness of F, so perfectly indistinguishable, thus $\mathsf{Hyb}_{3.1} \equiv \mathsf{Hyb}_{3.2}$. Observe that the $\mathbf{M}_i$ multiplication that Γ applies now cancels with $\mathbf{M}_i^{-1}$ and this is also true for the dual $\mathcal{D}$, because we sample $\mathbf{A}_i \cdot \mathbf{M}_i^{-1}$, and then when we multiply by $\mathbf{M}_i$ in the the computation of $\mathbf{T}$ this cancels again.

- Now we get to Hyb_4, in which we stop applying the permutation Γ to the output of Π (and likewise stop applying Γ^{-1} to the input of Π^{-1}), and also stop sampling $\mathbf{A}_i$ multiplied by $\mathbf{M}_i^{-1}$. The subspace $T_i^{\perp}$ is simply defined as $\mathsf{ColSpan}\left(\mathbf{A}_i^{(1)}\right)^{\perp}$, i.e., $\widetilde{\mathbf{T}}_i := \mathbf{A}_i^{(1)}$. Due to our explanation of the cancellation of the matrix multiplication $\mathbf{M}_i$, this is the same as what happens in $\mathsf{Hyb}_{3.2}$, so perfectly indistinguishable.

It follows that the success probability ϵ_4 in Hyb_4 equals the success probability from the previous hybrid.

- Hyb_5: Moving back to using an exponential number of cosets, by using small-range distribution again.

We rewind the process of sampling an R-small range distribution version of F, and use F as a standard random function. By the same argument for the indistinguishability between Hyb_0 and Hyb_1, the output of the current process has statistical distance bounded by $\frac{300\cdot q^3}{R} = \frac{\epsilon}{2^7\cdot k^2}$, which means in particular that the outputs of this hybrid and the previous hybrid has statistical distance bounded by $\frac{\epsilon}{2^7\cdot k^2}$. It follows that the success probability of the current hybrid is

$$:= \epsilon_5 \geq \epsilon_4 - \frac{\epsilon}{2^7\cdot k^2} \geq \frac{3\cdot\epsilon}{2^7\cdot k^2} - \frac{\epsilon}{2^7\cdot k^2} = \frac{\epsilon}{2^6\cdot k^2} \ .$$

To conclude, note that the process Hyb_5 is exactly the process where $\mathcal{A}$ executes on input oracle sampled from $\left(\mathcal{P},\mathcal{P}^{-1},\mathcal{D}'\right) \leftarrow \mathcal{O}'_{n,r,k,s}$. This finishes our proof. □

3.2 Simulating the Dual

In this section we prove the following lemma.

Lemma 24. *Suppose there is an oracle aided q-query quantum algorithm $\mathcal{A}$ such that*

$$\Pr\left[\begin{array}{l} y_0 = y_1 := y, \\ (\mathbf{u}_0 - \mathbf{u}_1) \notin \mathsf{ColSpan}\left(\mathbf{A}(y)^{(1)}\right) \end{array} : \begin{array}{rl} \left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}'\right) & \leftarrow \mathcal{O}'_{n,r,k,s} \\ (x_0, x_1) & \leftarrow \mathcal{A}^{\mathcal{P},\mathcal{P}^{-1},\mathcal{D}'} \\ (y_b, \mathbf{u}_b) & \leftarrow \mathcal{P}(x_b) \end{array}\right] \geq \epsilon \ .$$

Then, there is an oracle aided q-query quantum algorithm $\mathcal{B}$ such that

$$\Pr\left[(\overline{y}_0 = \overline{y}_1) \wedge (\overline{x}_0 \neq \overline{x}_1) \ : \begin{array}{rl} \left(\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}, \overline{\mathcal{D}}\right) & \leftarrow \mathcal{O}_{r+s,\, r,\, k-(n-r-s)} \\ (\overline{x}_0, \overline{x}_1) & \leftarrow \mathcal{B}^{\overline{\mathcal{P}},\overline{\mathcal{P}}^{-1}} \\ (\overline{y}_b, \overline{\mathbf{u}}_b) & \leftarrow \overline{\mathcal{P}}\left(\overline{x}_b\right) \end{array}\right] \geq \epsilon \ .$$

Proof. We first describe the actions of the algorithm $\mathcal{B}$ (which will use the code of $\mathcal{A}$ as part of its machinery) and then argue why it breaks collision resistance with the appropriate probability. Given oracle access to $\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}$ which comes from $\left(\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}, \overline{\mathcal{D}}\right) \leftarrow \mathcal{O}_{r+s,\, r,\, k-(n-r-s)}$, the algorithm $\mathcal{B}$ does the following:

- Sample a random function $F_{\mathbf{C}}$ that outputs a uniformly random invertible $\mathbf{C}(y) \in \mathbb{Z}_2^{k\times k}$, and a uniformly random vector $\mathbf{d}(y) \in \mathbb{Z}_2^{n-r-s}$. Sample a random n-bit permutation Γ. Define the following oracles.
- $\left(\ y \in \mathbb{Z}_2^r,\ \mathbf{u} \in \mathbb{Z}_2^k\ \right) \leftarrow \mathcal{P}\left(x \in \mathbb{Z}_2^n\right)$:
  - $\left(\ \overline{x} \in \mathbb{Z}_2^{r+s},\ \widetilde{x} \in \mathbb{Z}_2^{n-r-s}\ \right) \leftarrow \Gamma(x)$.
  - $\left(\ y \in \mathbb{Z}_2^r,\ \overline{\mathbf{u}} \in \mathbb{Z}_2^{k-(n-r-s)}\ \right) \leftarrow \overline{\mathcal{P}}(\overline{x})$.
  - $\left(\ \mathbf{C}(y) \in \mathbb{Z}_2^{k\times k},\ \mathbf{d}(y) \in \mathbb{Z}_2^{n-r-s}\right) \leftarrow F_{\mathbf{C}}(y)$.
  - $\mathbf{u} \leftarrow \mathbf{C}(y) \cdot \begin{pmatrix} \overline{\mathbf{u}} \\ \widetilde{x} + \mathbf{d}(y) \end{pmatrix}$.
- $\left(\ x \in \mathbb{Z}_2^n\ \right) \leftarrow \mathcal{P}^{-1}\left(\ y \in \mathbb{Z}_2^r,\ \mathbf{u} \in \mathbb{Z}_2^k\ \right)$:
  - $\left(\ \mathbf{C}(y) \in \mathbb{Z}_2^{k\times k},\ \mathbf{d}(y) \in \mathbb{Z}_2^{n-r-s}\right) \leftarrow F_{\mathbf{C}}(y)$.
  - $\begin{pmatrix} \overline{\mathbf{u}} \\ \widetilde{x} \end{pmatrix} \leftarrow \mathbf{C}(y)^{-1} \cdot \mathbf{u} - \begin{pmatrix} 0^{k-(n-r-s)} \\ \mathbf{d}(y) \end{pmatrix}$.
  - $\left(\ \overline{x} \in \mathbb{Z}_2^{r+s}\ \right) \leftarrow \overline{\mathcal{P}}^{-1}\left(y, \overline{\mathbf{u}}\right)$.
  - $x \leftarrow \Gamma^{-1}\left(\overline{x}, \widetilde{x}\right)$.
- $\mathbf{c}_{y,\mathbf{v}} \leftarrow \mathcal{D}'\left(\ y \in \mathbb{Z}_2^r,\ \mathbf{v} \in \mathbb{Z}_2^k\ \right)$:
  - $\left(\ \mathbf{C}(y) \in \mathbb{Z}_2^{k\times k},\ \mathbf{d}(y) \in \mathbb{Z}_2^{n-r-s}\right) \leftarrow F_{\mathbf{C}}\left(y\right)$.
  - $\mathbf{A}^{(1)}(y) :=$ last $n-r-s$ columns of $\mathbf{C}(y)$.
  - Simulate the answer using $\mathbf{A}^{(1)}(y)$, which is sufficient.

The remainder of the reduction is simple: $\mathcal{B}$ executes $(x_0, x_1) \leftarrow \mathcal{A}^{\mathcal{P},\mathcal{P}^{-1},\mathcal{D}'}$ and then $(\overline{x}_b, \widetilde{x}_b) \leftarrow \Gamma(x_b)$ and outputs $(\overline{x}_0, \overline{x}_1)$. Assume that the output of $\mathcal{A}$ satisfies $y_0 = y_1 := y$ and also $(\mathbf{u}_0 - \mathbf{u}_1) \notin \mathsf{ColSpan}\left(\mathbf{A}(y)^{(1)}\right)$, and recall that $\mathbf{A}(y)^{(1)} \in \mathbb{Z}_2^{k\times(n-r-s)}$ are the last $n-r-s$ columns of the matrix $\mathbf{A}(y) \in \mathbb{Z}_2^{k\times(n-r)}$, which is generated by the reduction. We explain why it is necessarily the case that $\overline{x}_0 \neq \overline{x}_1$.

First note that due to how we defined the reduction, $\mathbf{A}(y) := \mathbf{C}(y) \cdot \begin{pmatrix} \overline{\mathbf{A}}(y) & \\ & \mathbf{I}_{n-r-s} \end{pmatrix}$, where $\overline{\mathbf{A}}(y) \in \mathbb{Z}_2^{(k-(n-r-s))\times s}$ is the matrix arising from the oracles $\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}$ and $\mathbf{I}_{n-r-s} \in \mathbb{Z}_2^{(n-r-s)\times(n-r-s)}$ is the identity matrix of dimension $n-r-s$. Also note that because $\mathbf{C}(y)$, $\overline{\mathbf{A}}(y)$ are full rank then $\mathbf{A}(y)$ is full rank. Now, since $(\mathbf{u}_0 - \mathbf{u}_1) \notin \mathsf{ColSpan}\left(\mathbf{A}(y)^{(1)}\right)$ and since $\mathbf{A}(y)^{(1)}$ are the last $n-r-s$ columns of $\mathbf{A}(y)$, it follows that if we consider the coordinates vector $\mathbf{x} \in \mathbb{Z}_2^{n-r}$ of $(\mathbf{u}_0 - \mathbf{u}_1)$ with respect to $\mathbf{A}(y)$, the first s elements are not 0^s. By linearity of matrix multiplication it follows that if we look at each of the two coordinates vectors $\mathbf{x}_0$, $\mathbf{x}_1$ (each has $n-r$ bits) for $\mathbf{u}_0$, $\mathbf{u}_1$, respectively, somewhere in the first s bits, they differ. Now, recall how we obtain the first s bits of $\mathbf{x}_b$ – this is exactly by applying $\overline{\Pi}$ (the permutation on $\{0,1\}^{r+s}$ arising from the oracles $\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}$) to $\overline{x}_b$ and taking the last s bits of the output. Since these bits differ in the output of the permutation, then the preimages have to differ, i.e., $\overline{x}_0 \neq \overline{x}_1$.

Define $\epsilon_{\mathcal{B}}$ as the probability that the output of $\mathcal{A}$ indeed satisfies $y_0 = y_1 := y$ and also $(\mathbf{u}_0 - \mathbf{u}_1) \notin \mathsf{ColSpan}\left(\mathbf{A}^{(1)}(y)\right)$, and it remains to give a lower bound for the probability $\epsilon_{\mathcal{B}}$. We do this by a sequence of hybrids, eventually showing that the oracle which $\mathcal{B}$ simulates to $\mathcal{A}$ is indistinguishable from an oracle sampled from $\mathcal{O}'_{n,r,k,s}$. More precisely, each hybrid describes a process, it has an output, and a success predicate on the output.

- Hyb_0: The above distribution $\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}'\right) \leftarrow \mathcal{B}^{\overline{\mathcal{P}},\overline{\mathcal{P}}^{-1}}$, simulated to the algorithm $\mathcal{A}$.

The first hybrid is where $\mathcal{B}$ executes $\mathcal{A}$ by the simulation described above. The output of the process is the output (x_0, x_1) of $\mathcal{A}$. The process execution is considered as successful if $y_0 = y_1 := y$ and $(\mathbf{u}_0 - \mathbf{u}_1) \notin \mathsf{ColSpan}\left(\mathbf{A}(y)^{(1)}\right)$.

- Hyb_1: Not applying the inner permutation $\overline{\Pi}$ (which comes from the oracles $\overline{\mathcal{P}}$, $\overline{\mathcal{P}}^{-1}$), by using the random permutation Γ.

Let $\overline{\Pi}$ the permutation on $\{0,1\}^{r+s}$ that's inside $\overline{\mathcal{P}}$. In the previous hybrid we apply the n-bit permutation Γ to the input $x \in \mathbb{Z}_2^n$ and then proceed to apply the inner permutation $\overline{\Pi}$ to the first (i.e. leftmost) $r+s$ output bits of the first permutation Γ (we also apply Γ^{-1} to the output of the inverse of the inner permutation, in the inverse oracle $\mathcal{P}^{-1}$). The change we make to the current hybrid is that we simply apply only Γ and discard the inner permutation and its inverse. Since a random permutation concatenated with any permutation distributes identically to a random permutation, the current hybrid is statistically equivalent to the previous and in particular the output of this process distributes identically to the output of the previous, and so does the success probability.

- Hyb_2: For every $y \in \mathbb{Z}_2^r$, taking $\mathbf{A}(y)$ to be the direct output of F, by using the randomness of the random function.

In order to describe the change between the current and previous hybrid we first recall the structure of the oracles from the previous hybrid: Observe that in the previous hybrid, for every $y \in \mathbb{Z}_2^r$ we defined $\mathbf{A}(y) := \mathbf{C}(y) \cdot \begin{pmatrix} \overline{\mathbf{A}}(y) & \\ & \mathbf{I}_{n-r-s} \end{pmatrix}$, where $\mathbf{C}(y) \in \mathbb{Z}_2^{k \times k}$ is the output of $F_{\mathbf{C}}(y)$ and $\overline{\mathbf{A}}(y) \in \mathbb{Z}_2^{(k-(n-r-s)) \times s}$ is the output of the inner random function $\overline{F}$ (which comes from the inside of the oracles $\left(\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}\right)$). In the current hybrid we are going to ignore the inner random function $\overline{F}$, its generated matrix $\overline{\mathbf{A}}(y)$ and also the pair $\mathbf{C}(y)$, $\mathbf{d}(y)$, sample a fresh random function $F_{\mathbf{A}}$ at the beginning of the process, and on query y generate $(\mathbf{A}(y), \mathbf{b}(y)) \leftarrow F_{\mathbf{A}}(y)$, for $\mathbf{A}(y) \in \mathbb{Z}_2^{k \times (n-r)}$, $\mathbf{b}(y) \in \mathbb{Z}_2^k$.

To see why the two distributions are indistinguishable, note that the following two ways to sample $\mathbf{A}(y)$, are statistically equivalent: (1) For every $y \in \mathbb{Z}_2^r$, the matrix $\mathbf{A}(y)$ is generated by sampling a random full-rank matrix $\mathbf{C}(y) \in \mathbb{Z}_2^{k \times k}$ and letting $\mathbf{A}(y)$ be $\mathbf{C}(y) \cdot \begin{pmatrix} \overline{\mathbf{A}}(y) & \\ & \mathbf{I}_{n-r-s} \end{pmatrix}$. (2) For every $y \in \mathbb{Z}_2^r$ just sample a full-rank matrix $\mathbf{A}(y) \in \mathbb{Z}_2^{k \times (n-r)}$. Since we are using random functions and in the previous hybrid we are sampling $\mathbf{A}(y)$ according to (1) and in the current hybrid we are sampling $\mathbf{A}(y)$ according to (2), the outputs of the two hybrids distribute identically.

Finalizing the reduction. Observe that the distribution generated in the above Hyb_2 is exactly an oracle sampled from $\mathcal{O}'_{n,r,k,s}$. From the lemma's assumptions, the success probability for Hyb_2 is thus ϵ. Since we also showed that the hybrids have identical success probabilities, it follows that $\epsilon_{\mathcal{B}} = \epsilon$, which finishes our proof. □

We conclude this section by stating the following Theorem, which is obtained as a direct corollary from Lemmas 23 and 24.

Theorem 25. *Suppose there is an oracle aided q-query quantum algorithm $\mathcal{A}$ such that*

$$\Pr\left[x_0 \neq x_1 \wedge H(x_0) = H(x_1) \ : \ \begin{matrix} (\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}) & \leftarrow \mathcal{O}_{n,r,k} \\ (x_0, x_1) & \leftarrow \mathcal{A}^{\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}} \end{matrix}\right] \geq \epsilon \ .$$

Also, let $s \in \mathbb{N}$ such that $s \leq n - r - \lambda$, define $s' := s - (n - r - s)$, and assume

1. $\frac{\lambda \cdot k^8 \cdot q^8 \cdot s}{\sqrt{2^{n-r-\lambda-s}}} \leq o\left(\epsilon^4\right)$,
2. $\frac{k^2 \cdot q^3 \cdot (n-r-s)}{2^{s'}} \leq o\left(\epsilon^2\right)$.

Then, there is an oracle aided q-query quantum algorithm $\mathcal{B}$ such that

$$\Pr\left[x_0 \neq x_1 \wedge H(x_0) = H(x_1) \ : \ \begin{matrix} (\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}) \leftarrow \mathcal{O}_{r+s,\, r,\, k-(n-r-s)} \\ (x_0, x_1) \leftarrow \mathcal{B}^{\mathcal{P}, \mathcal{P}^{-1}} \end{matrix}\right] \geq \frac{\epsilon}{2^6 \cdot k^2} \ .$$

3.3 Hardness of the Dual-free Case from Claw-free Permutations

In this section we show that collision-finding in the dual-free case is provably hard with respect to an oracle. Our proof is different than the proof of [SZ25] for collision resistance, which is based on random 2-to-1 functions, and we refer the reader to the technical overview for an intuitive explanation on the difference.

We define coset partition functions (a notion defined in [SZ25]) and claw-free permutations, two objects we will use in the proof.

Definition 26 (Coset Partition Functions)**.** *For $n, \ell \in \mathbb{N}$ such that $\ell \leq n$ we say a function $Q : \{0,1\}^n \to \{0,1\}^m$ is a (n, m, ℓ)-*coset partition function *if, for each y in the image of Q, the pre-image set $Q^{-1}(y)$ has size 2^ℓ and is a coset of a linear space of dimension ℓ. We allow different pre-image sets to be cosets of different linear spaces.*

Definition 27 (Claw-free Permutation)**.** *For security parameter $\lambda \in \mathbb{N}$, a sample from the distribution of claw-free permutations for parameter λ is given as follows. Sample two uniformly random permutations $\Pi_0, \Pi_1 : \{0,1\}^\lambda \to \{0,1\}^\lambda$ and let $H^* : \{0,1\}^{\lambda+1} \to \{0,1\}^\lambda$ the function that for input $\left(b \in \{0,1\}, x \in \{0,1\}^\lambda\right)$ outputs $\Pi_b(x)$. The output sample of the claw-free distribution is H^*.*

From Dual-free to Claw-free Permutations. We next show that finding collisions in the dual-free case is at least as hard as finding collisions in claw-free permutations, while crucially keeping the dimension that the cosets live in, denoted by k, linear in the security parameter. Specifically, in the below Theorem, k need not be as large as n and only requires $k \geq n - r + \lambda$, which is a key point of difference compared to the reduction from [SZ25].

Theorem 28 (Reduction to Collision-finding in Claw-free Permutations)**.** *Let $n, r, k \in \mathbb{N}$ with $n > r$ and furthermore such that $\frac{n}{n-r}$ is an integer (which implies that $\frac{r}{n-r}$ is also an integer). Denote $\lambda := \frac{r}{n-r}$ (which implies $\lambda + 1 = \frac{n}{n-r}$) and additionally assume $k \geq n - r + \lambda$. Suppose there is an oracle aided q-query quantum algorithm $\mathcal{A}$ such that*

$$\Pr\left[(y_0 = y_1) \wedge (x_0 \neq x_1) \ : \ \begin{array}{rl} (\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}) & \leftarrow \mathcal{O}_{n,r,k} \\ (x_0, x_1) & \leftarrow \mathcal{A}^{\mathcal{P}, \mathcal{P}^{-1}} \\ (y_b, \mathbf{u}_b) & \leftarrow \mathcal{P}(x_b) \end{array}\right] \geq \epsilon \ .$$

Then there is an oracle aided q-query quantum algorithm $\mathcal{B}$ that given oracle access to $H^* : \{0,1\}^{\lambda+1} \to \{0,1\}^\lambda$, a uniformly random claw-free permutation, satisfies

$$\Pr\left[(H^*(w_0) = H^*(w_1)) \wedge (w_0 \neq w_1) \ : \ (w_0, w_1) \leftarrow \mathcal{B}^{H^*}\right] \geq \frac{\epsilon}{n-r} \ .$$

Proof. The reduction $\mathcal{B}$ works in two main steps. First, it turns oracle access to H^* into oracle access to some $(n, r, n-r)$-coset partition function Q with a special property. Then, it will turn oracle access to Q into oracle access to $\mathcal{P}, \mathcal{P}^{-1}$, while crucially keeping the parameter k small.

$\mathcal{B}$ samples $n-r-1$ instances of claw-free functions $H_1, \cdots, H_{n-r-1}$, and for each it also samples the "trapdoor" i.e. for every instance $i \in [n-r-1]$ it samples not only the permutations $\Pi_{i,0}, \Pi_{i,1} : \{0,1\}^\lambda \to \{0,1\}^\lambda$, but also the inverses $\Pi_{i,0}^{-1}, \Pi_{i,1}^{-1} : \{0,1\}^\lambda \to \{0,1\}^\lambda$. Next, sample a uniformly random place $i^* \in [n-r]$ and consider the parallel repetition function $Q : \{0,1\}^n \to \{0,1\}^r$, defined as follows.

$$Q\left(\mathbf{w} \in \{0,1\}^n\right) := H'_1\left(\mathbf{w}_1\right), \cdots, H'_{n-r}\left(\mathbf{w}_{n-r}\right) \ ,$$

where H^* is in place i^*, i.e., $H'_{i^*} = H^*$, and for every $i \in [n-r]$, $\mathbf{w}_i$ is derived by partitioning $\mathbf{w} \in \{0,1\}^n$ into consecutive, equally-sized $n-r$ parts, each one (accordingly) consisting of $\frac{n}{n-r} := \lambda + 1$ bits. We will later explain why Q is a $(n, r, n-r)$-coset partition function and furthermore has an additional useful property of having "foldable" cosets.

First, we continue to describe the reduction. $\mathcal{B}$ chooses a random permutation $\Gamma : \{0,1\}^n \to \{0,1\}^n$, and for each $y \in \{0,1\}^r$, it chooses a random full-column-rank matrix $\mathbf{C}(y) \in \mathbb{Z}_2^{k \times (n-r+\lambda)}$ (which is possible since we assume $k \geq n - r + \lambda$) and random vector $\mathbf{d}(y) \in \mathbb{Z}_2^k$. It then runs $\mathcal{A}$, simulating the oracles $\mathcal{P}, \mathcal{P}^{-1}$ as follows:

The oracle $\mathcal{P}\left(x \in \{0,1\}^n\right)$:

1. Compute $\mathbf{w} \leftarrow \Gamma(x)$.
2. $y \leftarrow Q(\mathbf{w})$.
3. **Fold $\mathbf{w} \in \{0,1\}^n$ into $\widetilde{\mathbf{w}} \in \{0,1\}^{n-r+\lambda}$:**
   (a) Generate a coordinates vector $\mathbf{r} \in \mathbb{Z}_2^{n-r}$ such that for $i \in [n-r]$, bit i of $\mathbf{r}$ is the first bit of $\mathbf{w}_i \in \{0,1\}^{\lambda+1}$, which in turn is the input to H'_i inside the computation of Q.
   (b) For each $\mathbf{w}_i \in \{0,1\}^{\lambda+1}$ consider $(\mathbf{w}_i)_{-1} \in \{0,1\}^\lambda$, derived by discarding the first bit of $\mathbf{w}_i$. Then set
   $$\widetilde{\mathbf{w}} := \left(\mathbf{r}, \sum_{i \in [n-r]} (\mathbf{w}_i)_{-1}\right) \quad .$$
4. Output $\left(y, \mathbf{C}(y) \cdot \widetilde{\mathbf{w}} + \mathbf{d}(y)\right)$.

The oracle $\mathcal{P}^{-1}\left(y \in \{0,1\}^r, \mathbf{u} \in \mathbb{Z}_2^k\right)$:

1. $x \leftarrow \begin{cases} \Gamma^{-1}\left(\mathsf{Unfold}\left(\widetilde{\mathbf{w}}, y\right)\right) & \exists\, \widetilde{\mathbf{w}} \in \mathbb{Z}_2^{n-r+\lambda} \text{ such that } \mathbf{C}(y) \cdot \widetilde{\mathbf{w}} + \mathbf{d}(y) = \mathbf{u} \\ \perp & \text{if no such } \widetilde{\mathbf{w}} \text{ exists} \end{cases}$
2. $\mathbf{w} \leftarrow \mathsf{Unfold}\left(\widetilde{\mathbf{w}}, y\right)$ executes as follows.
   (a) Recall the index $i^* \in [n-r]$ such that $H'_{i^*} := H^*$, where H^* is the claw-free permutation which the reduction is trying to break. Also recall that for every $j \in [n-r] \setminus \{i^*\}$ we have the inverse permutations $\Pi_{j,0}^{-1}, \Pi_{j,1}^{-1}$, as the reduction $\mathcal{B}$ sampled them by itself.
   (b) Parse $\widetilde{\mathbf{w}} := \left(\mathbf{r} \in \mathbb{Z}_2^{n-r}, \overline{\mathbf{w}} \in \{0,1\}^\lambda\right)$. Write $y = (y_1, \cdots, y_{n-r})$ where $y_i \in \{0,1\}^{\frac{r}{n-r}}$ for every $i \in [n-r]$.
   (c) For every $j \in [n-r] \setminus \{i^*\}$ compute $\mathbf{w}_{(j,\, y_j,\, \mathbf{r}_j)} \leftarrow \Pi_{j,\mathbf{r}_j}^{-1}(y_j)$, where $\mathbf{r}_j \in \{0,1\}$ is the j-th bit of $\mathbf{r}$.
   (d) Subtract to get
   $$\mathbf{w}^* \leftarrow \overline{\mathbf{w}} - \sum_{j \in [n-r] \setminus \{i^*\}} \mathbf{w}_{(j,\, y_j,\, \mathbf{r}_j)} \quad .$$
   (e) Output
   $$\mathbf{w} \leftarrow \Bigg(\left(\mathbf{r}_1, \mathbf{w}_{(1,\, y_1,\, \mathbf{r}_1)}\right), \cdots, \left(\mathbf{r}_{i^*-1}, \mathbf{w}_{\left(i^*-1,\, y_{i^*-1},\, \mathbf{r}_{i^*-1}\right)}\right), \left(\mathbf{r}_{i^*}, \mathbf{w}^*\right), \left(\mathbf{r}_{i^*+1}, \mathbf{w}_{\left(i^*+1,\, y_{i^*+1},\, \mathbf{r}_{i^*+1}\right)}\right), \cdots, \left(\mathbf{r}_{n-r}, \mathbf{w}_{(n-r,\, y_{n-r},\, \mathbf{r}_{n-r})}\right) \Bigg) \quad .$$

3. Output $\begin{cases} x & \text{if } x \neq \perp \text{ and } Q\left(\Gamma(x)\right) = y \\ \perp & \text{if } x = \perp \text{ or } Q(\Gamma(x)) \neq y \end{cases}$

Before we explain the rationale behind the above reduction, we observe a number of properties of the function Q.

Q is a coset partition function. First, it can be verified by the reader that since the functions H'_i are all 2-to-1, then Q is a $(n, r, n-r)$-coset partition function. Let us be more formal about this: every output $y \in \{0,1\}^r$ of Q is given by $n-r$ outputs $y_i \in \{0,1\}^\lambda$ of the functions H'_i, each of these consists of $\lambda := \frac{r}{n-r}$ bits. Since H'_i is 2-to-1, for every y_i there exist a pair $\mathbf{w}_{(i,\, y_i,\, 0)}, \mathbf{w}_{(i,\, y_i,\, 1)} \in \{0,1\}^{\lambda+1}$ such that $H'_i\left(\mathbf{w}_{(i,\, y_i,\, 0)}\right) = H'_i\left(\mathbf{w}_{(i,\, y_i,\, 1)}\right) = y_i$.

We can extend each of $\mathbf{w}_{(i,\, y_i,\, 0)}, \mathbf{w}_{(i,\, y_i,\, 1)} \in \{0,1\}^{\lambda+1}$ to $\mathbf{w}'_{(i,\, y_i,\, 0)}, \mathbf{w}'_{(i,\, y_i,\, 1)} \in \{0,1\}^n$ by padding with zeros, and placing the original $\mathbf{w}_{(i,\, y_i,\, 0)}, \mathbf{w}_{(i,\, y_i,\, 1)}$ at packet $i \in [n-r]$. Specifically, the resulting $\mathbf{w}'_{(i,\, y_i,\, b)} \in \{0,1\}^n$ will ultimately contain $n-r$ packets, each of size $\lambda + 1 = \frac{n}{n-r}$ and all but a single one (i.e., $n-r-1$ packets) will contain only zeros, and the single non-zero packet will be at index $i \in [n-r]$, which will contain the original $\mathbf{w}_{(i,\, y_i,\, b)} \in \{0,1\}^{\lambda+1}$.

Now, observe what are the cosets generated by Q: For every $y \in \{0,1\}^r$ consider the matrix $\overline{\mathbf{A}}_y \in \mathbb{Z}_2^{n \times (n-r)}$ such that for every $i \in [n-r]$, its column i is $\mathbf{w}'_{(i,\, y_i,\, 0)} + \mathbf{w}'_{(i,\, y_i,\, 1)}$. Also, consider the vector $\overline{\mathbf{b}}_y \in \{0,1\}^n$ defined as $\overline{\mathbf{b}}_y := \sum_{i \in [n-r]} \mathbf{w}'_{(i,\, y_i,\, 0)}$. It remains to observe that for every $y \in \{0,1\}^r$ output of Q, the set of inputs that maps to y is given by the coset $\mathsf{ColSpan}\left(\overline{\mathbf{A}}_y\right) + \overline{\mathbf{b}}_y$.

For every preimage coset in Q there exists a folded coset, with reversible mapping. We showed that for every output y of Q, its preimage set $Q^{-1}(y)$ is a coset of the form $\mathsf{ColSpan}\left(\overline{\mathbf{A}}_y\right) + \overline{\mathbf{b}}_y$ for some $\overline{\mathbf{A}}_y \in \mathbb{Z}_2^{n \times (n-r)}$, $\overline{\mathbf{b}}_y \in \mathbb{Z}_2^n$. We next show that while the preimage sets of the outputs of Q might need to be large (i.e. their elements take n bits), for every y there also exists *a folded coset* $\mathsf{ColSpan}\left(\widetilde{\mathbf{A}}_y\right) + \widetilde{\mathbf{b}}_y$ for some $\widetilde{\mathbf{A}}_y \in \mathbb{Z}_2^{(n-r+\lambda) \times (n-r)}$, $\widetilde{\mathbf{b}}_y \in \mathbb{Z}_2^{n-r+\lambda}$, and not only that, the reduction $\mathcal{B}$ above shows how to go *reversibly* between the original and folded cosets.

The folded coset is defined as follows. For every y_i consider the pair $\mathbf{w}_{(i,\, y_i,\, 0)}, \mathbf{w}_{(i,\, y_i,\, 1)} \in \{0,1\}^{\lambda+1}$ such that $H'_i\left(\mathbf{w}_{(i,\, y_i,\, 0)}\right) = H'_i\left(\mathbf{w}_{(i,\, y_i,\, 1)}\right) = y_i$. Let $\widetilde{\mathbf{A}}_y \in \mathbb{Z}_2^{(n-r+\lambda) \times (n-r)}$ the matrix such that its i-th column has two parts. The first part takes $n-r$ bits and is equal to the i-th column of the identity matrix $\mathbf{I}_{(n-r) \times (n-r)}$. The remaining λ bits are equal to $\left(\mathbf{w}_{(i,\, y_i,\, 0)} + \mathbf{w}_{(i,\, y_i,\, 1)}\right)_{-1}$. As for the coset shift $\widetilde{\mathbf{b}}_y \in \mathbb{Z}_2^{n-r+\lambda}$, it also has two parts. The first part is 0^{n-r}, and the second part is $\sum_{i \in [n-r]} \left(\mathbf{w}_{(i,\, y_i,\, 0)}\right)_{-1}$.

We conclude with two observations. First, the efficient and reversible mapping between the cosets: to fold $\mathbf{w} \to \widetilde{\mathbf{w}}$ one executes Step 3 in the reduction's simulation of $\mathcal{P}$, and to unfold, one needs the information of the image $y \in \{0,1\}^r$ and executes Step 2 in the above simulation of $\mathcal{P}^{-1}$. Finally, a key property which lets all of this connect, is that for any preimage $\mathbf{w}$ such that $Q(\mathbf{w}) = y$, the coordinates vector $\mathbf{z} \in \mathbb{Z}_2^{n-r}$ of $\mathbf{w}$ in the original coset $\mathsf{ColSpan}\left(\overline{\mathbf{A}}_y\right) + \overline{\mathbf{b}}_y$ and its folding $\widetilde{\mathbf{w}} \in \mathsf{ColSpan}\left(\widetilde{\mathbf{A}}_y\right) + \widetilde{\mathbf{b}}_y$, are the same. Specifically, one can observe that this is exactly the vector $\mathbf{r} \in \mathbb{Z}_2^{n-r}$. Thus, when going back and forth between the cosets using the reduction $\mathcal{B}$, the coordinates $\mathbf{z}$ do not change.

Correct distribution of the simulated oracles $\mathcal{P}, \mathcal{P}^{-1}$. In a nutshell, what we'll do is use both cosets, the original and the folded one, to correctly simulate the two corresponding parts of the oracles $\mathcal{P}, \mathcal{P}^{-1}$, which are the random permutation Π inside and the random cosets $\mathsf{ColSpan}\left(\mathbf{A}(y)\right) + \mathbf{b}(y)$ for every output y. The original coset $\mathsf{ColSpan}\left(\overline{\mathbf{A}}_y\right) + \overline{\mathbf{b}}_y$ will be used to simulate the permutation Π, and the folded coset $\mathsf{ColSpan}\left(\widetilde{\mathbf{A}}_y\right) + \widetilde{\mathbf{b}}_y$ will be used to simulate the cosets $\mathsf{ColSpan}\left(\mathbf{A}(y)\right) + \mathbf{b}(y)$, i.e., obliviously sample $\mathbf{A}(y) \in \mathbb{Z}_2^{k \times (n-r)}$ and $\mathbf{b}(y) \in \mathbb{Z}_2^k$.

We start with showing that the simulated $\mathcal{P}$ distributes correctly. We define an augmented function $Q' : \{0,1\}^n \to \{0,1\}^n$. On input $\mathbf{w}$, the n-bit output of $Q'(\mathbf{w})$ consists of two parts. The first r bits are set to $y = Q(\mathbf{w})$. The preimage set $Q^{-1}(y)$ is then a coset, which can be described as the set $\{\overline{\mathbf{A}}_y \cdot \mathbf{r} + \overline{\mathbf{b}}_y\}$ as $\mathbf{r}$ ranges over $\mathbb{Z}_2^{n-r}$ (where $\overline{\mathbf{A}}_y$, $\overline{\mathbf{b}}_y$ are both unknown to the reduction algorithm $\mathcal{B}$), as we described above. Define the function $\overline{J}(\mathbf{w})$ that outputs the unique vector in $\mathbb{Z}_2^{n-r}$ such that $\mathbf{w} = \overline{\mathbf{A}}_y \cdot \overline{J}(\mathbf{w}) + \overline{\mathbf{b}}_y$. Then define $Q'(\mathbf{w}) = \left(Q(\mathbf{w}), \overline{J}(\mathbf{w})\right)$. Note that Q' is not efficiently computable without knowing $\overline{\mathbf{A}}_y$,$\overline{\mathbf{b}}_y$, but here we will only need it to exist, and not need it to be efficiently computable. Notice that Q' is a function from $\mathbb{Z}_2^n$ to $\mathbb{Z}_2^n$, and it is moreover a permutation with $(Q')^{-1}(y, \mathbf{r}) = \overline{\mathbf{A}}_y \cdot \mathbf{r} + \overline{\mathbf{b}}_y$.

Since for every vector $\mathbf{w}$ with $Q(\mathbf{w}) = y$, its coordinates vector $\overline{J}(\mathbf{w}) \in \mathbb{Z}_2^{n-r}$ is the same in both, the original coset $\mathsf{ColSpan}\left(\overline{\mathbf{A}}_y\right) + \overline{\mathbf{b}}_y$, and the folded coset $\mathsf{ColSpan}\left(\widetilde{\mathbf{A}}_y\right) + \widetilde{\mathbf{b}}_y$, observe that $\mathcal{B}$'s simulation of $\mathcal{P}$ is implicitly setting the following parameters

$$\Pi(x) = Q'(\Gamma(x)) \;, \qquad H(x) = Q(\Gamma(x)) \;, \qquad J(x) = \overline{J}(\Gamma(x)) \;,$$
$$\mathbf{A}(y) = \mathbf{C}(y) \cdot \widetilde{\mathbf{A}}_y \;, \qquad \mathbf{b}(y) = \mathbf{C}(y) \cdot \widetilde{\mathbf{b}}_y + \mathbf{d}(y) \;.$$

Thus, we must check that these quantities have the correct distribution. Indeed, for every Q, which is in turn defines $\left(\overline{\mathbf{A}}_y, \overline{\mathbf{b}}_y\right)_{y \in \{0,1\}^r}$, the function $Q'(x)$ is a permutation: Given $y \in \{0,1\}^r$, $\mathbf{r} \in \mathbb{Z}_2^{n-r}$, one can recover $\mathbf{z} \in \{0,1\}^n$ as $\mathbf{z} = \overline{\mathbf{A}}_y \cdot \mathbf{r} + \overline{\mathbf{b}}_y$. Hence Π is a permutation since it is the composition of two permutations. Moreover, since one of the two permutations (Γ) is uniformly random, so is Π.

Now we look at the distribution of $\mathbf{A}(y), \mathbf{b}(y)$. Recall that $\widetilde{\mathbf{A}}_y \in \mathbb{Z}_2^{(n-r+\lambda) \times (n-r)}$ is a full-column-rank matrix, and $\mathbf{C}(y) \in \mathbb{Z}_2^{k \times (n-r+\lambda)}$ is a *random* full-column-rank matrix. Thus, $\mathbf{A}(y) = \mathbf{C}(y) \cdot \widetilde{\mathbf{A}}_y \in \mathbb{Z}_2^{k \times (n-r)}$ is also a random full-column-rank matrix. Also, we have that $\mathbf{b}(y) = \mathbf{C}(y) \cdot \widetilde{\mathbf{b}}_y + \mathbf{d}(y)$ where $\mathbf{d}(y)$ is random, meaning $\mathbf{b}(y)$ is random. Thus, $\mathcal{P}$ has an identical distribution to that arising from $\mathcal{O}_{n,r,k}$.

For the inverse $\mathcal{P}^{-1}$, observe that $\mathcal{P}^{-1}(\mathcal{P}(x)) = x$, and for all pairs $\left(y \in \{0,1\}^r, \mathbf{u} \in \mathbb{Z}_2^k\right)$ that are not in the image of $\mathcal{P}$, we have $\mathcal{P}^{-1}(y, \mathbf{u}) = \perp$. Thus, $\mathcal{P}^{-1}$ is the uniquely-defined inverse of $\mathcal{P}$. Thus, since the distribution of $\mathcal{P}$ simulated by $\mathcal{B}$ exactly matches the distribution arising from $\mathcal{O}_{n,r,k}$, the same is true of the pairs $\left(\mathcal{P}, \mathcal{P}^{-1}\right)$.

Finishing touches. We saw that for $H^* : \{0,1\}^{\lambda+1} \to \{0,1\}^\lambda$ a random claw-free permutation, the reduction $\mathcal{B}$ first simulates a foldable $(n, r, n-r)$-coset partition function Q, and proceeds to perfectly simulate the view of $\mathcal{A}$, which consists of the pair of oracles $\mathcal{P}, \mathcal{P}^{-1}$ that distribute according to $\mathcal{O}_{n,r,k}$. Hence, with probability ϵ, the algorithm $\mathcal{A}$ will produce a collision $x_0 \neq x_1$ such that $H(x_0) = H(x_1)$, where H is the first r output bits of $\mathcal{P}$. It remains to explain what $\mathcal{B}$ does in order to obtain a collision in H^*, for every collision in the simulated H. Given (x_0, x_1),

the reduction $\mathcal{B}$ will then compute and output $(\mathbf{w}_0 = \Gamma(x_0), \mathbf{w}_1 = \Gamma(x_1))$. Observe that if $x_0 \neq x_1$, then $\mathbf{w}_0 \neq \mathbf{w}_1$ since Γ is a permutation. Moreover, if $H(x_0) = H(x_1)$, then

$$Q(\mathbf{w}_0) = Q(\Gamma(x_0)) = H(x_0) = H(x_1) = Q(\Gamma(x_1)) = Q(\mathbf{w}_1) \quad .$$

Hence, with probability at least ϵ, $\mathcal{B}$ will output a collision for Q, which will constitute a collision $\mathbf{w}_{(i,\, y_i,\, 0)}, \mathbf{w}_{(i,\, y_i,\, 1)} \in \{0,1\}^{\lambda+1}$ in at least one H'_i. Finally, we chose at random $i^* \leftarrow [n-r]$, and for every choice of i^*, the distribution of the generated Q is identically distributed, so perfectly indistinguishable. Consequently, the probability that $i = i^*$ is at least $\frac{1}{n-r}$, so overall the probability that $\mathcal{B}$ gets out of $\mathcal{A}$ a collision is $\frac{\epsilon}{n-r}$. Notice that for each query that $\mathcal{A}$ makes to $\mathcal{P}(\cdot)$, the reduction $\mathcal{B}$ needs to make exactly one query to Q and the same goes for the inverse $\mathcal{P}^{-1}(\cdot)$. Each query to Q constitutes exactly one query to H^*. Thus $\mathcal{B}$ makes exactly q queries to H^*. This completes the proof. □

Collision-resistance of claw-free permutations with respect to an oracle. It remains to explain why a uniformly random claw-free permutation is collision resistant with respect to a quantumly-queriable classical oracle.

Lemma 29. *For $\lambda \in \mathbb{N}$, a random claw-free permutation $H : \{0,1\}^{\lambda+1} \to \{0,1\}^{\lambda}$ is collision resistant given quantum queries to H. In particular, any (unbounded) quantum algorithm making q queries has a $O\left(\frac{q^3}{2^\lambda}\right)$ probability of producing a collision.*

Proof. The sampling procedure of H is given by sampling two uniformly random permutation $\Pi_0, \Pi_1 : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$, and defining

$$H\left(b \in \{0,1\}, x \in \{0,1\}^{\lambda}\right) := \Pi_b(x) \quad .$$

We next recall that for every q-query quantum algorithm, quantum oracle access to $\Pi : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ a random permutation (but not its inverse) is $O\left(\frac{q^3}{2^\lambda}\right)$-indistinguishable from Π being a random function [Zha15]. This means that the overall H is now a random function from $\{0,1\}^{\lambda+1}$ to $\{0,1\}^{\lambda}$. We conclude with recalling that random functions are quantumly collision-resistant: The probability of finding a collision in such random H is at most $O\left(\frac{q^3}{2^\lambda}\right)$ by [Zha15]. Thus, we can bound the overall probability for finding a collision by $O\left(\frac{q^3}{2^\lambda}\right)$. □

4 Short One-Shot Signatures in the Standard Model

Following our construction in an oracle model from Section 3, we present our standard model constructions. As in the oracle model, we have our base Construction 30 of a non-collapsing CRH. Our standard model one-shot signature scheme and in particular its signing algorithm are identical to the oracle model, that is, it uses the base construction in a black box way, in the same way that the oracle model OSS uses the oracle model base construction.

Construction 30. *Let $\lambda \in \mathbb{N}$ the statistical security parameter. Define $s := 18 \cdot \lambda$ and let $n, r, k \in \mathbb{N}$ such that $r := s \cdot \lambda$, $n := r + \frac{3}{2} \cdot s$, $k := \frac{3}{2} \cdot s + \lambda$. Let $d := \mathsf{poly}_d(\lambda) \in \mathbb{N}$ the expansion parameter and $\kappa := \mathsf{poly}_\kappa(\lambda) \in \mathbb{N}$ the cryptographic security parameter, for some sufficiently large polynomials in the statistical security parameter.*

Let iO *an iO scheme,* $(\mathsf{F}, \mathsf{Punc})$ *a puncturable PRF, and* $\left(\Pi, \Pi^{-1}, \mathsf{Permute}\right)$ *a permutable PRP for the class of all* $\left(2^{\mathsf{poly}(\kappa)}, \mathsf{poly}(\kappa)\right)$*-decomposable permutations. Then we construct a hash function* $(\mathsf{Gen}, \mathsf{Hash})$ *as follows:*

- $\mathsf{Gen}\left(1^{\lambda}\right)$*: Sample* $k_{\mathsf{in}}, k_{\mathsf{out}}, k_{\mathsf{lin}} \leftarrow \{0,1\}^{\kappa}$*.* $\Pi\left(k_{\mathsf{in}}, \cdot\right)$ *is a permutation with domain* $\{0,1\}^{n}$*,* $\Pi\left(k_{\mathsf{out}}, \cdot\right)$ *is a permutation with domain* $\{0,1\}^{d}$*, and* $\mathsf{F}\left(k_{\mathsf{lin}}, \cdot\right)$ *is a PRF with inputs in* $\{0,1\}^{d}$*. Let* $H(\cdot)$ *denote the first* r *output bits of* $\Pi\left(k_{\mathsf{in}}, \cdot\right)$ *and* $J(\cdot)$ *denote the remaining* $n-r$ *bits. For each* $y \in \{0,1\}^{d}$*, using the output pseudorandomness of* $\mathsf{F}\left(k_{\mathsf{lin}}, y\right)$ *we sample* $\mathbf{A}(y) \in \mathbb{Z}_2^{k \times (n-r)}$ *and* $\mathbf{b}(y) \in \mathbb{Z}_2^{k}$*. The matrix* $\mathbf{A}(y) \in \mathbb{Z}_2^{k \times (n-r)}$ *is with full column rank* $n-r$*, and also such that its bottom* λ *rows have full row rank* λ*.* $\mathbf{b}(y) \in \mathbb{Z}_2^{k}$ *is a pseudorandom vector.*

  As the common reference string output $\mathsf{CRS} = \left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}\right)$ *where* $\mathcal{P} \leftarrow \mathsf{iO}\left(1^{\kappa}, P\right)$*,* $\mathcal{P}^{-1} \leftarrow \mathsf{iO}\left(1^{\kappa}, P^{-1}\right)$*,* $\mathcal{D} \leftarrow \mathsf{iO}\left(1^{\kappa}, D\right)$ *such that*

$$P(x) = \left(\ y \ , \ \mathbf{A}(y) \cdot J(x) + \mathbf{b}(y) \ \right) \text{ where } y \leftarrow \Pi^{-1}\left(k_{\mathsf{out}}, \ H(x) || 0^{d-r}\right)$$

$$P^{-1}\left(y, \mathbf{u}\right) = \begin{cases} \Pi^{-1}\left(w || \mathbf{z}\right) & \exists \ w, \mathbf{z} : \left(\Pi\left(k_{\mathsf{out}}, y\right) = w || 0^{d-r}\right) \wedge \left(\mathbf{A}(y) \cdot \mathbf{z} + \mathbf{b}(y) = \mathbf{u}\right) \\ \bot & \text{else} \end{cases}$$

$$\mathcal{D}\left(y, \mathbf{v}\right) = \begin{cases} \mathbf{c}_{y,\mathbf{v}} & \text{if } \mathbf{v}^{T} \cdot \mathbf{A}(y) \in \mathsf{RowSpan}\left(\mathbf{A}(y)_{[(k-\lambda)+1]}, \cdots, \mathbf{A}(y)_{[(k-\lambda)+\lambda]}\right) \\ \bot & \text{otherwise} \end{cases}$$

  where for $\mathbf{A} \in \mathbb{Z}_2^{k \times (n-r)}$ *and* $j \in [k]$*,* $\mathbf{A}_{[j]} \in \mathbb{Z}_2^{n-r}$ *is the* j*-th row of* $\mathbf{A}$ *(e.g.,* k*-th row is bottom), and* $\mathbf{c}_{y,\mathbf{v}} \in \mathbb{Z}_2^{\lambda}$ *is the coordinates vector, of the vector* $\mathbf{v}^{T} \cdot \mathbf{A}(y) \in \mathbb{Z}_2^{n-r}$ *with respect to the basis* $\mathbf{A}(y)_{[(k-\lambda)+1]}, \cdots, \mathbf{A}(y)_{[(k-\lambda)+\lambda]}$ *(i.e., element* $j \in [\lambda]$ *of* $\mathbf{c}_{y,\mathbf{v}}$ *is the coefficient of* $\mathbf{A}(y)_{[(k-\lambda)+j]}$*).*

- $\mathsf{Hash}\left(\mathsf{CRS}, x\right)$*: Compute* $(y, \mathbf{u}) \leftarrow \mathcal{P}(x)$ *and output* y*.*

Security in the Standard Model. In this section we prove the main Theorems 2, 3, 4. As in the oracle model, we need to prove the collision resistance of H in order to prove strong unforgeability of our OSS, which we will focus on for the rest of this section.

4.1 Bloating the Dual

For $n', r', k' \in \mathbb{N}$, $s \in \mathbb{N} \cup \{0\}$ such that $s \leq n' - r' - \lambda$, we define the modified generator $\widetilde{\mathsf{Gen}}\left(1^{\lambda}, n', r', k', s\right)$, as follows. It samples a distribution over $\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}'$, where the values of "$n, r, k$" in Construction 30 are replaced by n', r', k' respectively. For s we let $\mathbf{A}(y)^{(0)} \in \mathbb{Z}_2^{k' \times s}$ denote the first s columns of $\mathbf{A}(y) \in \mathbb{Z}_2^{k' \times (n'-r')}$ and $\mathbf{A}(y)^{(1)} \in \mathbb{Z}_2^{k' \times (n'-r'-s)}$ denote the remaining $n'-r'-s$ columns. Note that the standard generator is defined as $\widetilde{\mathsf{Gen}}\left(1^{\lambda}, n, r, k, 0\right)$. In the the setting of $\widetilde{\mathsf{Gen}}\left(1^{\lambda}, n, r, k, s\right)$, the functionality of $\mathcal{P}, \mathcal{P}^{-1}$ stays the same, and for the dual oracle, we do the same things as in the original $\mathcal{D}$, but with respect to $\mathbf{A}(y)^{(1)} \in \mathbb{Z}_2^{k \times (n-r-s)}$ rather than with

respect to $\mathbf{A}(y) \in \mathbb{Z}_2^{k\times(n-r)}$. Formally:

$$\mathcal{D}'(y,\mathbf{v}) = \begin{cases} \mathbf{c}_{y,\mathbf{v}} & \text{if } \mathbf{v}^T \cdot \mathbf{A}(y)^{(1)} \in \mathsf{RowSpan}\left(\mathbf{A}(y)^{(1)}_{[(k-\lambda)+1]}, \cdots, \mathbf{A}(y)^{(1)}_{[(k-\lambda)+\lambda]}\right) \\ \perp & \text{otherwise} \end{cases}$$

where here, similarly to the construction (but not quite the same), $\mathbf{c}_{y,\mathbf{v}}$ is the coordinates vector, of the vector $\mathbf{v}^T \cdot \mathbf{A}(y)^{(1)} \in \mathbb{Z}_2^{n-r-s}$ with respect to the basis $\mathbf{A}(y)^{(1)}_{[(k-\lambda)+1]}, \cdots, \mathbf{A}(y)^{(1)}_{[(k-\lambda)+\lambda]}$.

Lemma 31. *Let $\lambda, n, r, k \in \mathbb{N}$, and assume there is a quantum algorithm $\mathcal{A}$ with complexity $T_{\mathcal{A}}$ such that,*

$$\Pr\left[(y_0 = y_1) \wedge (x_0 \neq x_1) \ : \ \begin{matrix} (\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}) & \leftarrow \widetilde{\mathsf{Gen}}\left(1^\lambda, n, r, k, 0\right) \\ (x_0, x_1) & \leftarrow \mathcal{A}\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}\right) \\ (y_b, \mathbf{u}_b) & \leftarrow \mathcal{P}\left(x_b\right) \end{matrix}\right] \geq \epsilon \ .$$

For our primitives in Construction 30, let the iO scheme be $\left(f_{\mathsf{iO}}(\cdot), \frac{1}{f_{\mathsf{iO}}(\cdot)}\right)$-secure, the puncturable PRF be $\left(f_{\mathsf{PRF}}(\cdot), \frac{1}{f_{\mathsf{PRF}}(\cdot)}\right)$-secure, the permutable PRP be $\left(f_{\mathsf{PRP}}(\cdot), \frac{1}{f_{\mathsf{PRP}}(\cdot)}\right)$-secure and assume that Lossy Functions (as in Definition 9) are $\left(f_{\mathsf{LF}}(\cdot), \frac{1}{f_{\mathsf{LF}}(\cdot)}\right)$-secure. Also assume that OWFs are $\left(f_{\mathsf{OWF}}(\cdot), \frac{1}{f_{\mathsf{OWF}}(\cdot)}\right)$-secure for $f_{\mathsf{OWF}}(\lambda) := 2^{\lambda^\delta}$ for some constant real number $\delta > 0$ and let $w := \lambda^{\frac{\delta}{2}}$, $s' := s - (n - r - s)$.

Assume $f_{\mathsf{iO}}(\kappa), f_{\mathsf{PRF}}(\kappa), f_{\mathsf{PRP}}(\kappa), f_{\mathsf{LF}}(w) \geq T_{\mathcal{A}} + \mathsf{poly}'(\lambda)$ for some fixed polynomial $\mathsf{poly}'(\cdot)$ and all of the following:

1. *For $f := \min(f_{\mathsf{iO}}, f_{\mathsf{PRF}}, f_{\mathsf{PRP}})$, $\frac{2^r}{f(\kappa)} \leq \frac{\epsilon}{k^2 \cdot 512}$,*
2. *$\frac{1}{f_{\mathsf{LF}}(w)} \leq \frac{\epsilon}{k^2 \cdot 256}$,*
3. *$\frac{2^{2\cdot w} \cdot k^4 \cdot \left(k^3 + T_{\mathcal{A}}\right)}{f_{\mathsf{OWF}}(n-r-\lambda-s) - \mathsf{poly}(n-r-\lambda-s)} \leq \epsilon^2$, and*
4. *$\frac{4\cdot(n-r-s)\cdot 2^w}{2^{s'}} \leq \epsilon$.*

Then, it follows that,

$$\Pr\left[\begin{matrix} y_0 = y_1 := y, \\ (\mathbf{u}_0 - \mathbf{u}_1) \notin \mathsf{ColSpan}\left(\mathbf{A}(y)^{(1)}\right) \end{matrix} \ : \ \begin{matrix} (\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}') & \leftarrow \widetilde{\mathsf{Gen}}(1^\lambda, n, r, k, s) \\ (x_0, x_1) & \leftarrow \mathcal{A}\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}'\right) \\ (y_b, \mathbf{u}_b) & \leftarrow \mathcal{P}(x_b) \end{matrix}\right] \geq \frac{\epsilon}{512 \cdot k^2} \ .$$

Note that $\mathbf{u}_0 - \mathbf{u}_1 \notin \mathsf{ColSpan}\left(\mathbf{A}(y)^{(1)}\right)$ means in particular that $\mathbf{u}_0, \mathbf{u}_1$, and hence x_0, x_1, are distinct. Thus, the second expression means that $\mathcal{A}$ is finding collisions, but these collisions satisfy an even stronger requirement.

Proof. Let $\lambda \in \mathbb{N}$ and assume there is a $T_{\mathcal{A}}$-complexity algorithm $\mathcal{A}$ and a probability ϵ such that $\mathcal{A}$ gets $\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}\right) \leftarrow \mathsf{Gen}\left(1^\lambda\right)$ and outputs a pair (x_0, x_1) of n-bit strings such that with probability ϵ we have $x_0 \neq x_1$ and $y_0 = y_1$. We next define a sequence of hybrid experiments. Each hybrid defines a computational process, an output of the process and a predicate computed on the process output. The predicate defines whether the (hybrid) process execution was successful or not.

- Hyb_0: The original execution of $\mathcal{A}$.

The process Hyb_0 is the above execution of $\mathcal{A}$ on input a sample from the distribution $\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}\right) \leftarrow \widetilde{\mathsf{Gen}}\left(1^\lambda, n, r, k, 0\right)$. We define the output of the process as (x_0, x_1) and the process execution is considered as successful if x_0, x_1 are both distinct and collide in their y values. By definition, the success probability of Hyb_0 is ϵ.

- Hyb_1: Preparing to switch to a bounded number of cosets $(\mathbf{A}(y), \mathbf{b}(y))$, by using an obfuscated puncturable PRF and injective mode of a lossy function.

Let $(\mathsf{LF.KeyGen}, \mathsf{LF.F})$ a $\left(f(\cdot), \frac{1}{f(\cdot)}\right)$-secure lossy function scheme (as in Definition 9). Sample $\mathsf{pk}_{\mathsf{LF}} \leftarrow \mathsf{LF.KeyGen}\left(1^d, 0, 1^w\right)$ where w is as defined in our Lemma statement, and let $\mathsf{LF.F}\left(\mathsf{pk}_{\mathsf{LF}}, \cdot\right) : \{0,1\}^d \rightarrow \{0,1\}^m$ the induced injective function.

We now consider two circuits in order to describe our current hybrid. $E_0\left(k_{\mathsf{lin}}, \cdot\right)$ is the circuit that given an input from $\{0,1\}^d$ applies $\mathsf{F}\left(k_{\mathsf{lin}}, \cdot\right)$ to get $\mathbf{A}(y)$, $\mathbf{b}(y)$. $E_1\left(\mathsf{pk}_{\mathsf{LF}}, k'_{\mathsf{lin}}, \cdot\right)$ is the circuit that for a LF key $\mathsf{pk}_{\mathsf{LF}}$ and P-PRF key k'_{lin} for a P-PRF with input size m rather than d, given a d-bit input y, applies the lossy function with key $\mathsf{pk}_{\mathsf{LF}}$ and then the P-PRF to get $\mathbf{A}'(y)$, $\mathbf{b}'(y)$.

Note that in the previous hybrid, the circuit E_0 is used in all three circuits P, P^{-1}, D in order to generate the cosets per input y, and furthermore, each of these three circuits access E_0 only as a black box. The change that we make to the current hybrid is that we are going to use $E_1\left(\mathsf{pk}_{\mathsf{LF}}, k'_{\mathsf{lin}}, \cdot\right)$ for a freshly sampled $\mathsf{pk}_{\mathsf{LF}}$, k'_{lin}, instead of E_0. Observe that in both cases, whether we are using E_0 or E_1, for each value in $\{0,1\}^r$ if we use real randomness to sample the coset, the distributions of the cosets are statistically equivalent between the two cases. Since we are sending obfuscations $\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}\right)$ of the three circuits and due to the three circuits accessing the samplers E_0, E_1 only as black boxes, it follows by Lemma 13 that the output of the previous hybrid and the current hybrid are $\left(f(\kappa), \frac{|\mathcal{X}|}{f(\kappa)}\right)$-indistinguishable, where $f := \min\left(f_{\mathsf{iO}}, f_{\mathsf{PRF}}\right)$ and $\mathcal{X}$ is the set of all possible values of y, which has size 2^r (recall that while the $y's$ are of length $d >> r$, these are a sparse set inside $\{0,1\}^d$ because we are padding with zeros), and we also recall that κ, the cryptographic security parameter is some polynomial in the statistical security parameter λ. It follows by Condition 1 from our Lemma statement, that the success probability of the current process is $:= \epsilon_1 \geq \epsilon - \frac{2^r}{f(\kappa)} \geq \epsilon - \frac{\epsilon}{32} = \frac{31 \cdot \epsilon}{32}$.

- Hyb_2: Switching to a bounded number of cosets $(\mathbf{A}(y), \mathbf{b}(y))$, by using the lossy function.

The change from the previous hybrid to the current hybrid is that we are going to sample a lossy key $\mathsf{pk}^1_{\mathsf{LF}} \leftarrow \mathsf{LF.KeyGen}\left(1^d, 1, 1^w\right)$ and use it inside E_1 from the previous hybrid, instead of using an injective key $\mathsf{pk}^0_{\mathsf{LF}} \leftarrow \mathsf{LF.KeyGen}\left(1^d, 0, 1^w\right)$, which was used in the previous hybrid. Note that in this hybrid, there are at most 2^w cosets (that is, some different values of y will have the same coset), by the correctness of the lossy function scheme. By the security of the lossy function scheme, the output of this hybrid is $\left(f_{\mathsf{LF}}(w), \frac{1}{f_{\mathsf{LF}}(w)}\right)$-indistinguishable from the previous hybrid. It follows by Condition 2 from our Lemma statement that the success probability of the current process is

$$:= \epsilon_2 \geq \epsilon_1 - \frac{1}{f_{\mathsf{LF}}(w)} \geq \frac{31 \cdot \epsilon}{32} - \frac{\epsilon}{32} \geq \frac{30 \cdot \epsilon}{32} \quad .$$

- Hyb_3: Relaxing dual verification oracle to accept a larger subspace, by using an obfuscated puncturable PRF and subspace hiding functions.

The change we make in this hybrid is that we make the membership check dual oracles more relaxed. Formally, we invoke Lemma 14 with the following interface, for every value $m_y := \mathsf{LF.F}(\mathsf{pk}_{\mathsf{LF}}, y)$. The smallest oracle "$S_0$" from the Lemma 14 statement is simply $S_{y,0}^{\perp} := \mathsf{ColSpan}(\mathbf{A}(y))^{\perp}$ here and "S" from the lemma is $S_y^{\perp}$, defined as the total set of vectors that $\mathcal{D}$ does not output $\perp$ on. The partition "P" from the lemma statement is the partition defined by the output of the oracle $\mathcal{D}$ (observe that this is indeed a partition into cosets of S that are all parallel to S_0). That is, the oracle $\mathcal{D}$ computes the function F_S for these subspaces.

Now, for every y, after sampling $(\mathbf{A}(y), \mathbf{b}(y))$, we sample superspace $T_{y,0}^{\perp}$ and by the statement of Lemma 14, (1) $T_{y,0}^{\perp}$ is a random superspace of $S_{y,0}^{\perp}$ (which itself has dimension $k-(n-r)$) with $k-(n-r-s)$ dimensions, and (2) $T_y^{\perp}$ is the joint span of $T_{y,0}^{\perp}$ and $S_y^{\perp}$. For concreteness and because it will later be useful to our proof, assume that our process of sampling $T_{y,0}^{\perp}$ is by sampling its dual $T_{y,0}$, which in turn is sampled by sampling a uniformly random invertible $\mathbf{M}(y) \in \mathbb{Z}_2^{(n-r)\times(n-r)}$, multiplying by $\mathbf{A}(y)$ and then taking the rightmost $n-r-s$ columns of the generated matrix. We denote by C_y the circuit which gives access to the function F_S and by C'_y the circuit that gives access to the F_T. It follows there is a process that takes a value y, computes $m_y := \mathsf{LF.F}(\mathsf{pk}_{\mathsf{LF}}, y)$ and than there is a process (which is accessed in a black-box fashion) that computes $C_y := C_{m_y}$ as a function of m_y alone.

To argue indistinguishability formally, we now consider two circuits in order to describe our current hybrid. The first circuit $E_S(k_S, \cdot)$ is the circuit that given an input m_y computes C_y and returns an obfuscation of it. The second circuit $E_T(k_T, \cdot)$ is the circuit that given an input m_y samples C'_y and returns an obfuscation of it. In the previous hybrid, in order to compute the output of $\mathcal{D}$ we used E_S and in the current hybrid we switch to using E_T.

Let $\mathcal{X}$ the set of all possible values m_y that arise from the scheme, which has size $\leq 2^w$ by the lossy function. Per such value, by Lemma 14 we have $\left(f_{\mathsf{OWF}}(n-r-\lambda-s), \frac{s}{f_{\mathsf{OWF}}(n-r-\lambda-s)}\right)$-indistinguishability between using E_S or E_T on input y, and this happens when the circuits (and the random algebraic objects inside them) are sampled with truly random bits. It follows by Lemma 13 that the output of the previous hybrid and the current hybrid are $\left(f_{\mathsf{OWF}}(n-r-\lambda-s), \frac{|\mathcal{X}|}{f_{\mathsf{OWF}}(n-r-\lambda-s)}\right)$-indistinguishable, where $\mathcal{X}$ is the set of all cosets derived from our scheme, which has size $\leq 2^w$ by the lossy function. It follows by Condition 3 that the success probability of the current process is

$$:= \epsilon_3 \geq \epsilon_2 - \frac{2^w}{f_{\mathsf{OWF}}(n-r-\lambda-s)} \geq \frac{30\cdot\epsilon}{32} - \frac{\epsilon}{32} = \frac{29\cdot\epsilon}{32} \quad .$$

- Hyb_4: Asking for sum of collisions to be outside of $T_{y,0}$, using the hardness of concentration in dual of obfuscated subspace, the pseudorandomness of the puncturable PRF and the security of iO.

This hybrid is the same as the previous in terms of execution, but we change the definition of a successful execution, that is, we change the predicate computed on the output of the process. We still ask that $(y_0 = y_1 := y)$, but instead of only asking the second requirement to be $(x_0 \neq x_1)$, we ask for a stronger condition: $(\mathbf{u}_0 - \mathbf{u}_1) \in (S_{y,0} \setminus T_{y,0})$. Note that we are not going to need to be able to efficiently check for the success of the condition, but we'll prove that it happens with a good probability nonetheless.

Let ϵ_4 be the success probability of the current hybrid and note that $\mathcal{A}$ finds collisions with probability ϵ_3 in the previous hybrid (and since this hybrid is no different and only change the

definition of a successful execution, the same goes for the current hybrid). Let $\mathcal{X} \subseteq \{0,1\}^m$ the image of the lossy function $\mathsf{LF.F}(\mathsf{pk}_{\mathsf{LF}}, \cdot)$ which we use to map our images y to cosets $(\mathbf{A}(y), \mathbf{b}(y))$, that is, there are at most $|\mathcal{X}|$ cosets and by the lossyness we know that $|\mathcal{X}| \leq 2^w$. For every value $\mathbf{x} \in \mathcal{X}$ denote by $\epsilon_3^{(\mathbf{x})}$ the probability to find a collision on value $\mathbf{x}$, or formally, that $y_0 = y_1 := y$, $x_0 \neq x_1$ and $\mathbf{x} = \mathsf{LF.F}(\mathsf{pk}_{\mathsf{LF}}, y)$. We deduce $\sum_{\mathbf{x}\in\mathcal{X}} \epsilon_3^{(\mathbf{x})} = \epsilon_3$. Let L be a subset of $\mathcal{X}$ such that $\epsilon_3^{(\mathbf{x})} \geq \frac{\epsilon_3}{2\cdot|\mathcal{X}|}$ and note that $\sum_{\mathbf{x}\in L} \epsilon_3^{(\mathbf{x})} \geq \frac{\epsilon_3}{2}$. Similarly, we define $\epsilon_4^{(\mathbf{x})}$ as the probability to find a "strong" collision on value $\mathbf{x}$, or formally, that $y_0 = y_1 := y$, $(\mathbf{u}_0 - \mathbf{u}_1) \in (S_{y,0} \setminus T_{y,0})$ and $\mathbf{x} = \mathsf{LF.F}(\mathsf{pk}_{\mathsf{LF}}, y)$. Note that $S_{y,0}$, $T_{y,0}$ are really functions of $\mathbf{x}$ rather than of y, so we can denote $(\mathbf{u}_0 - \mathbf{u}_1) \in (S_{\mathbf{x},0} \setminus T_{\mathbf{x},0})$ and also observe that $\sum_{\mathbf{x}\in\mathcal{X}} \epsilon_4^{(\mathbf{x})} = \epsilon_4$.

We would now like to use Lemma 16, so we make sure that we satisfy its requirements. Let any $\mathbf{x} \in L$, we know that by definition, $\epsilon_3^{(\mathbf{x})} \geq \frac{\epsilon_3}{2\cdot|\mathcal{X}|}$ and also recall that $\epsilon_3 \geq \frac{28\cdot\epsilon}{32}$, $|\mathcal{X}| \leq 2^w$ and thus

$$\epsilon_3^{(\mathbf{x})} \geq \frac{\epsilon_3}{2\cdot|\mathcal{X}|} \geq \frac{29\cdot\epsilon}{64}\cdot\frac{1}{2^w} > \frac{\epsilon}{2\cdot 2^w} \quad .$$

Let $s' := s - (n-r-s)$ and for any $\mathbf{x} \in L$ let $\ell_{\mathbf{x}} := \frac{k^2}{\epsilon_3^{(\mathbf{x})}} \leq \frac{k^2\cdot 2^w\cdot 2}{\epsilon}$. Note that by our Lemma 31 statement's Condition 4 we have (1) $\frac{(n-r-s)}{2^{s'}} \leq \frac{\epsilon_3^{(\mathbf{x})}}{2}$, and by Condition 3 we have (2) $\ell_{\mathbf{x}}\cdot\left(k^3 + T_{\mathcal{A}}\right) \leq \frac{f_{\mathsf{OWF}}(n-r-\lambda-s)-\mathsf{poly}(n-r-\lambda-s)}{2\cdot\lambda\cdot\ell_{\mathbf{x}}}$. Since this satisfies Lemma 16, it follows that for every $\mathbf{x} \in L$ we have $\epsilon_4^{(\mathbf{x})} \geq \frac{\epsilon_3^{(\mathbf{x})}}{16\cdot k^2} - \frac{1}{f(\kappa)}$ for $f := \min(f_{\mathsf{iO}}, f_{\mathsf{PRF}})$. The expression $\frac{1}{f(\kappa)}$ is subtracted because for each $\mathbf{x} \in \mathcal{X}$, in order to use Lemma 16, we need the randomness for the experiment to be genuinely random, which will necessitate us to invoke the security of the iO and puncturable PRF, which incurs the loss of $\frac{1}{f(\kappa)}$. It follows that

$$\epsilon_4 = \sum_{\mathbf{x}\in\mathcal{X}} \epsilon_4^{(\mathbf{x})} \geq \sum_{\mathbf{x}\in L} \epsilon_4^{(\mathbf{x})} \geq \sum_{\mathbf{x}\in L} \frac{\epsilon_3^{(\mathbf{x})}}{16\cdot k^2} - \frac{|\mathcal{X}|}{f(\kappa)} \geq \frac{\left(\frac{\epsilon_3}{2}\right)}{16\cdot k^2} - \frac{|\mathcal{X}|}{f(\kappa)} \geq \frac{\left(\frac{29\cdot\epsilon}{64}\right)}{16\cdot k^2} - \frac{2^w}{f(\kappa)} \geq \frac{\epsilon}{64\cdot k^2} \quad .$$

- $\mathsf{Hyb}_{\bar{5}}$: For every y, de-randomizing $T_{y,0}$ and defining it as the column span of $\mathbf{A}(y)^{(1)} \in \mathbb{Z}_2^{k\times(n-r-s)}$, the last $n-r-s$ columns of the matrix $\mathbf{A}(y)$, by using permutable PRPs, obfuscated puncturable PRF and the security of iO.

Recall that at the basis of our scheme there are the $\leq 2^w$ cosets $(\mathbf{A}(y), \mathbf{b}(y))_y$ and for each coset we also sample a pseudorandom subspace $T_{y,0}$. Recall that we sample a pseudorandom matrix $\mathbf{M}(y) \in \mathbb{Z}_2^{(n-r)\times(n-r)}$ which is explicitly used in the sampling of the subspace $T_{y,0}$, that is, we define $T_{y,0}$ as the column span of the rightmost $n-r-s$ columns of $\mathbf{A}(y)\cdot\mathbf{M}(y)$. The problem is that we use $\mathbf{A}(y), \mathbf{b}(y)$ as is in the oracles $\mathcal{P}$, $\mathcal{P}^{-1}$, but use $\mathbf{A}(y)\cdot\mathbf{M}(y)$ in $\mathcal{D}$. We will resolve this discrepancy in this hybrid. This hybrid is the same as the previous, with one change: For every y, after sampling the coset $(\mathbf{A}(y), \mathbf{b}(y))$, we will not continue to randomly sample $T_{y,0}^{\perp}$, and simply define $T_{y,0} := \mathsf{ColSpan}\left(\mathbf{A}(y)^{(1)}\right)$ such that $\mathbf{A}(y)^{(1)} \in \mathbb{Z}_2^{k\times(n-r-s)}$ is defined to be the last $n-r-s$ columns of the matrix $\mathbf{A}(y) \in \mathbb{Z}_2^{k\times(n-r)}$. We will define intermediate hybrids and then explain why the previous hybrid is indistinguishable from the current.

- **Using the security of the permutable PRP.** Assume we sample all components of our scheme, excluding the key k_{in} for the initial permutation Π on $\{0,1\}^n$. Define the following

permutation Γ on $\{0,1\}^n$: Denote by h the first r bits of the input and by j the last $n-r$ bits of the input to the permutation Γ. Recall that in the circuits P, P^{-1}, D, the value y, the coset $\mathbf{A}(y), \mathbf{b}(y)$, the superspace $T^{\perp}_{y,0}$ and the associated matrix $\mathbf{M}(y) \in \mathbb{Z}_2^{(n-r)\times(n-r)}$, are all computed as a function of r bits, which in the construction take the role of $H(x)$. In fact, all of the above variables can be written as a function of $h \in \{0,1\}^r$ instead of as a function of y.

The permutation Γ takes $h \in \{0,1\}^r$ and computes $\mathbf{M}(y)$, then interprets $j \in \{0,1\}^{n-r}$ as a vector in $\mathbb{Z}_2^{n-r}$, and applies $\mathbf{M}(y)$ to j. For every value $h \in \{0,1\}^r$ observe that multiplication by $\mathbf{M}(y)$ is a permutation (and moreover an affine permutation, which is decomposable efficiently). It follows that the overall Γ is a controlled version of an affine permutation, and thus is controlled on decomposable permutations. Overall, we deduce that Γ is a $\left(2^{\mathsf{poly}(\lambda)}, \mathsf{poly}(\lambda)\right)$-decomposable permutation. We use the permutable PRP Π, and switch to a setting where we use the key k^{Γ}_{in} that applies Γ to the output of Π (and Γ^{-1} to the input of Π^{-1}), instead of just applying Π and its inverse. By the security of the permutable PRPs, this change is $\left(f_{\mathsf{PRP}}(\kappa), \frac{1}{f_{\mathsf{PRP}}(\kappa)}\right)$-indistinguishable.

- **Using an obfuscated puncturable PRF and statistical equivalence of sampling the random matrix $\mathbf{A}(y)$, for every y.** In this intermediate hybrid, instead of sampling $\mathbf{A}(y)$ and then using it as is, we will sample it multiplied by the inverse of $\mathbf{M}(y)$, that is, our sampler gives us $\mathbf{A}(y) \cdot \mathbf{M}^{-1}(y)$ instead of $\mathbf{A}(y)$. Note that for every y, if we use a truly random $\mathbf{A}(y)$, the two processes are statistically equivalent because $\mathbf{M}$ is invertible. By a standard argument using an obfuscated puncturable PRF (i.e., we use lemma 13 and the fact that when using real randomness, the two ways to sample $\mathbf{A}(y)$ are statistically equivalent), we get that this change is $\left(f(\kappa), 2^w \cdot \frac{1}{f(\kappa)}\right)$-indistinguishable, for $f := \min\left(f_{\mathsf{iO}}, f_{\mathsf{PRF}}\right)$.

- **Using the security of outside iO.** We now can more easily see why the current hybrid is indistinguishable from the previous. Looking into the functionality, we see that the matrix multiplications which include $\mathbf{M}(y)$ all cancel out and can be ignored. Specifically, in the computation of $\mathcal{P}, \mathcal{P}^{-1}$, the concatenated permutation Γ effectively induce a computation by $\mathbf{M}(y)$, which then cancels out with the fact that $\mathbf{A}(y)$ is sampled with $\mathbf{A}(y) \cdot \mathbf{M}^{-1}(y)$. Inside the computation of $\mathcal{D}$, we sample $\mathbf{A}(y) \cdot \mathbf{M}^{-1}(y)$ from the PRF as in $\mathcal{P}, \mathcal{P}^{-1}$, and then when we make the multiplication by $\mathbf{M}(y)$ (which, as we recall, is used to derive the matrix $\mathbf{T}$ which in turn imply the identity of the subspace $T_{y,0}$), which cancels out again. Overall, the computation is logically equivalent to just sampling $\mathbf{A}(y)$ from the puncturable PRF, and using that in all oracles, in particular the $n-r-s$ rightmost columns of $\mathbf{A}(y)$ now yield $T_{y,0}$. Due to functional equivalence we can invoke the security of the outside iO and deduce that this change is $\left(f_{\mathsf{iO}}(\kappa), \frac{1}{f_{\mathsf{iO}}(\kappa)}\right)$-indistinguishable.

Observe that after the last change we are exactly in the setting of Hyb_5. It follows in particular that the success probability of the current process is

$$:= \epsilon_5 \geq \epsilon_4 - \frac{1}{f_{\mathsf{PRP}}(\kappa)} - 2^w \cdot \frac{1}{f(\kappa)} - \frac{1}{f_{\mathsf{iO}}(\kappa)} \geq \frac{\epsilon}{64 \cdot k^2} - \frac{\epsilon}{512 \cdot k^2} = \frac{7 \cdot \epsilon}{512 \cdot k^2} \ .$$

- Hyb_6: Going back to using 2^r cosets rather than $\leq 2^w$, by moving from lossy mode to injective mode in the lossy function.

We make the exact same change we made between Hyb_1 to Hyb_2, but in the opposite direction. That is we sample an injective key $\mathsf{pk}^0_{\mathsf{LF}} \leftarrow \mathsf{LF.KeyGen}\left(1^d, 0, 1^w\right)$ and use it instead of the previous lossy key $\mathsf{pk}^1_{\mathsf{LF}} \leftarrow \mathsf{LF.KeyGen}\left(1^d, 1, 1^w\right)$, which is used in the previous hybrid. By the exact same argument (which relies on the security of the lossy function), the output of this hybrid is $\left(f_{\mathsf{LF}}(w), \frac{1}{f_{\mathsf{LF}}(w)}\right)$-indistinguishable. It follows in particular the success probability of the current process is

$$:= \epsilon_6 \geq \epsilon_5 - \frac{1}{f_{\mathsf{LF}}(w)} \geq \frac{7 \cdot \epsilon}{512 \cdot k^2} - \frac{\epsilon}{256 \cdot k^2} = \frac{5 \cdot \epsilon}{512 \cdot k^2} \ .$$

- Hyb_7: Stop using the lossy function, by an obfuscated puncturable PRF.

We make the exact same change we made between Hyb_0 to Hyb_1, but in the opposite direction. That is, we drop the lossy function $\mathsf{LF.F}$ altogether and apply the PRF to y directly and not to $\mathbf{x}$, the output of the lossy function on input y. By the exact same argument (which relies on Lemma 13), the output of this hybrid is $\left(f(\kappa), \frac{|\mathcal{X}|}{f(\kappa)}\right)$-indistinguishable for $f := \min\left(f_{\mathsf{iO}}, f_{\mathsf{PRF}}\right)$, where $\mathcal{X}$ is the set of all possible values of y, which has size 2^r (recall that while the $y's$ are of length $d >> r$, these are a sparse set inside $\{0,1\}^d$ because we are padding with zeros). It follows that the success probability of the current process is

$$:= \epsilon_7 \geq \epsilon_6 - \frac{2^r}{f\left(\kappa\right)} \geq \frac{5 \cdot \epsilon}{512 \cdot k^2} - \frac{\epsilon}{512 \cdot k^2} = \frac{\epsilon}{128 \cdot k^2} \ .$$

To conclude, note that the generated obfuscations in the final hybrid form exactly the distribution $\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}'\right) \leftarrow \widetilde{\mathsf{Gen}}\left(1^\lambda, n, r, k, s\right)$. This finishes our proof. □

4.2 Simulating the Dual

Our next step is to show that an adversary which has access to the dual-free setting can simulate the CRS for an adversary in the restricted setting, where the dual verification check is bloated.

Lemma 32. *Let $\lambda, n, r, k \in \mathbb{N}$ and assume there is a quantum algorithm $\mathcal{A}$ running in time $T_{\mathcal{A}}$ such that,*

$$\Pr\left[\begin{array}{l} y_0 = y_1 := y, \\ \left(\mathbf{u}_0 - \mathbf{u}_1\right) \notin \mathsf{ColSpan}\left(\mathbf{A}(y)^{(1)}\right) \end{array} : \begin{array}{rl} \left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}'\right) & \leftarrow \widetilde{\mathsf{Gen}}(1^\lambda, n, r, k, s) \\ \left(x_0, x_1\right) & \leftarrow \mathcal{A}\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}'\right) \\ \left(y_b, \mathbf{u}_b\right) & \leftarrow \mathcal{P}(x_b) \end{array}\right] \geq \epsilon \ .$$

For our primitives in Construction 30, let the iO scheme be $\left(f_{\mathsf{iO}}\left(\cdot\right), \frac{1}{f_{\mathsf{iO}}(\cdot)}\right)$-secure, the puncturable PRF be $\left(f_{\mathsf{PRF}}\left(\cdot\right), \frac{1}{f_{\mathsf{PRF}}(\cdot)}\right)$-secure, the permutable PRP be $\left(f_{\mathsf{PRP}}\left(\cdot\right), \frac{1}{f_{\mathsf{PRP}}(\cdot)}\right)$-secure and let $f := \min\left(f_{\mathsf{iO}}, f_{\mathsf{PRF}}, f_{\mathsf{PRP}}\right)$. Assume $\frac{2^r}{f(\kappa)} \leq o\left(\epsilon\right)$. Then, there is a quantum algorithm $\mathcal{B}$ running in time $T_{\mathcal{A}} + \mathsf{poly}\left(\lambda\right)$ such that

$$\Pr\left[\left(\overline{y}_0 = \overline{y}_1\right) \wedge \left(\overline{x}_0 \neq \overline{x}_1\right) \ : \begin{array}{rl} \left(\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}, \overline{\mathcal{D}}\right) & \leftarrow \widetilde{\mathsf{Gen}}\left(1^\lambda, r+s, r, k-(n-r-s), 0\right) \\ \left(\overline{x}_0, \overline{x}_1\right) & \leftarrow \mathcal{B}\left(\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}\right) \\ \left(\overline{y}_b, \overline{\mathbf{u}}_b\right) & \leftarrow \overline{\mathcal{P}}(x_b) \end{array}\right] \geq \frac{\epsilon}{2} \ .$$

Proof. We first describe the actions of the algorithm $\mathcal{B}$ (which will use the code of $\mathcal{A}$ as part of its machinery) and then argue why it breaks collision resistance with the appropriate probability. Given $\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}$ which comes from $\left(\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}, \overline{\mathcal{D}}\right) \leftarrow \widetilde{\mathsf{Gen}}\left(1^\lambda, r+s, r, k-(n-r-s), 0\right)$, the algorithm $\mathcal{B}$ does the following:

- Sample a P-PRF key $k_{\mathbf{C}}$ that outputs some sufficient (polynomial) amount of random bits on an d-bit input, and sample a permutable PRP key k_Γ for a PRP on domain $\{0,1\}^n$. Define the following circuits.
- $\left(\ y \in \mathbb{Z}_2^d,\ \mathbf{u} \in \mathbb{Z}_2^k\ \right) \leftarrow P\left(x \in \mathbb{Z}_2^n\right)$:
  - $\left(\ \overline{x} \in \mathbb{Z}_2^{r+s},\ \widetilde{x} \in \mathbb{Z}_2^{n-r-s}\ \right) \leftarrow \Pi\left(k_\Gamma, x\right)$.
  - $\left(\ y \in \mathbb{Z}_2^d,\ \overline{\mathbf{u}} \in \mathbb{Z}_2^{k-(n-r-s)}\ \right) \leftarrow \overline{\mathcal{P}}(\overline{x})$.
  - $\left(\ \mathbf{C}(y) \in \mathbb{Z}_2^{k\times k},\ \mathbf{d}(y) \in \mathbb{Z}_2^{n-r-s}\right) \leftarrow \mathsf{F}\left(k_{\mathbf{C}}, y\right)$.
  - $\mathbf{u} \leftarrow \mathbf{C}(y) \cdot \begin{pmatrix} \overline{\mathbf{u}} \\ \widetilde{x} + \mathbf{d}(y) \end{pmatrix}$.
- $\left(\ x \in \mathbb{Z}_2^n\ \right) \leftarrow P^{-1}\left(\ y \in \mathbb{Z}_2^d,\ \mathbf{u} \in \mathbb{Z}_2^k\ \right)$:
  - $\left(\ \mathbf{C}(y) \in \mathbb{Z}_2^{k\times k},\ \mathbf{d}(y) \in \mathbb{Z}_2^{n-r-s}\right) \leftarrow \mathsf{F}\left(k_{\mathbf{C}}, y\right)$.
  - $\begin{pmatrix} \overline{\mathbf{u}} \\ \widetilde{x} \end{pmatrix} \leftarrow \mathbf{C}(y)^{-1} \cdot \mathbf{u} - \begin{pmatrix} 0^{k-(n-r-s)} \\ \mathbf{d}(y) \end{pmatrix}$.
  - $\left(\ \overline{x} \in \mathbb{Z}_2^{r+s}\ \right) \leftarrow \overline{\mathcal{P}}^{-1}\left(y, \overline{\mathbf{u}}\right)$.
  - $x \leftarrow \Pi^{-1}\left(k_\Gamma, (\overline{x}, \widetilde{x})\right)$.
- $\mathbf{c}_{y,\mathbf{v}} \leftarrow D'\left(\ y \in \mathbb{Z}_2^d,\ \mathbf{v} \in \mathbb{Z}_2^k\ \right)$:
  - $\left(\ \mathbf{C}(y) \in \mathbb{Z}_2^{k\times k},\ \mathbf{d}(y) \in \mathbb{Z}_2^{n-r-s}\right) \leftarrow \mathsf{F}\left(k_{\mathbf{C}}, y\right)$.
  - $\mathbf{A}(y)^{(1)} :=$ last $n-r-s$ columns of $\mathbf{C}(y)$.
  - Simulate the answer of D' using $\mathbf{A}(y)^{(1)}$, which is sufficient.
- Use indistinguishability obfuscation in order to generate the input for $\mathcal{A}$: $\mathcal{P} \leftarrow \mathsf{iO}\left(1^\kappa, P\right)$, $\mathcal{P}^{-1} \leftarrow \mathsf{iO}\left(1^\kappa, P^{-1}\right)$, $\mathcal{D}' \leftarrow \mathsf{iO}\left(1^\kappa, D'\right)$.

The remainder of the reduction is simple: $\mathcal{B}$ executes $(x_0, x_1) \leftarrow \mathcal{A}\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}'\right)$ and then $(\overline{x}_b, \widetilde{x}_b) \leftarrow \Pi\left(k_\Gamma, x_b\right)$ and outputs $(\overline{x}_0, \overline{x}_1)$. Assume that the output of $\mathcal{A}$ satisfies $y_0 = y_1 := y$ and also $(\mathbf{u}_0 - \mathbf{u}_1) \notin \mathsf{ColSpan}\left(\mathbf{A}(y)^{(1)}\right)$, and recall that $\mathbf{A}(y)^{(1)} \in \mathbb{Z}_2^{k\times(n-r-s)}$ are the last $n-r-s$ columns of the matrix $\mathbf{A}(y) \in \mathbb{Z}_2^{k\times(n-r)}$, which is generated by the reduction. We explain why it is necessarily the case that $\overline{x}_0 \neq \overline{x}_1$.

Why if $\mathcal{A}$ finds a collision then $\mathcal{B}$ finds a collision. First note that due to how we defined the reduction, $\mathbf{A}(y) := \mathbf{C}(y) \cdot \begin{pmatrix} \overline{\mathbf{A}}(y) & \\ & \mathbf{I}_{n-r-s} \end{pmatrix}$, where $\overline{\mathbf{A}}(y) \in \mathbb{Z}_2^{(k-(n-r-s))\times s}$ is the matrix arising from $\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}$ and $\mathbf{I}_{n-r-s} \in \mathbb{Z}_2^{(n-r-s)\times(n-r-s)}$ is the identity matrix of dimension $n-r-s$. Also note that because $\mathbf{C}(y)$, $\overline{\mathbf{A}}(y)$ are full rank then $\mathbf{A}(y)$ is full rank. Now, since $(\mathbf{u}_0 - \mathbf{u}_1) \notin \mathsf{ColSpan}\left(\mathbf{A}(y)^{(1)}\right)$ and since $\mathbf{A}(y)^{(1)}$ are the last $n-r-s$ columns of $\mathbf{A}(y)$, it follows that if we consider the coordinates vector $\mathbf{x} \in \mathbb{Z}_2^{n-r}$ of $(\mathbf{u}_0 - \mathbf{u}_1)$ with respect to $\mathbf{A}(y)$, the first s elements are not 0^s. By linearity of matrix multiplication it follows that if we look at each of the two coordinates vectors $\mathbf{x}_0$, $\mathbf{x}_1$ (each has $n-r$ bits) for $\mathbf{u}_0$, $\mathbf{u}_1$, respectively, somewhere in the first s bits, they differ. Now, recall how we obtain the first s bits of $\mathbf{x}_b$ – this is exactly by applying $\overline{\Pi}$ (the permutation on $\{0,1\}^{r+s}$ arising from $\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}$) to $\overline{x}_b$ and taking the last s bits of the output. Since these bits differ in the output of the permutation, then the preimages have to differ, i.e., $\overline{x}_0 \neq \overline{x}_1$.

Define $\epsilon_{\mathcal{B}}$ as the probability that the output of $\mathcal{A}$ indeed satisfies $y_0 = y_1 := y$ and also $(\mathbf{u}_0 - \mathbf{u}_1) \notin \mathsf{ColSpan}\left(\mathbf{A}(y)^{(1)}\right)$, and it remains to give a lower bound for the probability $\epsilon_{\mathcal{B}}$. We do this by a sequence of hybrids, eventually showing that the view which $\mathcal{B}$ simulates to $\mathcal{A}$ is computationally indistinguishable from a sample from $\widetilde{\mathsf{Gen}}\left(1^\lambda, n, r, k, s\right)$. More precisely, each hybrid describes a process, it has an output, and a success predicate on the output.

- Hyb_0: The above distribution $\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}'\right) \leftarrow \mathcal{B}\left(\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}\right)$, simulated to the algorithm $\mathcal{A}$.

The first distribution is defined in the reduction above. The output of the process is the output (x_0, x_1) of $\mathcal{A}$. The process execution is considered as successful if $y_0 = y_1 := y$ and $(\mathbf{u}_0 - \mathbf{u}_1) \notin \mathsf{ColSpan}\left(\mathbf{A}(y)^{(1)}\right)$.

- Hyb_1: Not applying the inner permutation $\overline{\Pi}_{\mathsf{in}}$ (which comes from the circuits $\overline{\mathcal{P}}$, $\overline{\mathcal{P}}^{-1}$), by using the security of an obfuscated permutable PRP.

Let $\overline{\Pi}_{\mathsf{in}}$ the (first) permutable PRP that's inside $\overline{\mathcal{P}}$ (which is the obfuscation of the circuit $\overline{P}$). In the previous hybrid we apply the n-bit permutable PRP $\Pi\left(k_\Gamma, \cdot\right)$ to the input $x \in \mathbb{Z}_2^n$ and then proceed to apply the inner permutation $\overline{\Pi}_{\mathsf{in}}\left(\overline{k}_{\mathsf{in}}, \cdot\right)$ to the first (i.e. leftmost) $r+s$ output bits of the first permutation $\Pi\left(k_\Gamma, \cdot\right)$. The change we make to the current hybrid is that we simply apply only $\Pi\left(k_\Gamma, \cdot\right)$.

Recall two details: (1) By the results of [SZ25] on permutable PRPs being decomposable, the inner permutable PRP $\overline{\Pi}_{\mathsf{in}}\left(\overline{k}_{\mathsf{in}}, \cdot\right)$ is in and of itself $\left(2^{\mathsf{poly}(\lambda)}, \mathsf{poly}(\lambda)\right)$-decomposable, and (2) the circuits P, P^{-1} which apply the permutations are both obfuscated by iO to be generate the obfuscations $\mathcal{P}$, $\mathcal{P}^{-1}$. We can treat it as a fixed permutation that acts on the output of the permutation $\Pi\left(k_\Gamma, \cdot\right)$ and thus it follows by Lemma by the results of [SZ25] that the current and previous hybrids are computationally indistinguishable, with indistinguishability $\frac{1}{f(\kappa)}$.

- Hyb_2: For every $y \in \mathbb{Z}_2^d$, taking $\mathbf{A}(y)$ to be the direct output of the PRF F, by using an obfuscated punctured PRF.

In order to describe the change between the current and previous hybrid we first recall the structure of the circuits from the previous hybrid: In the previous hybrid, for every $y \in \mathbb{Z}_2^r$ we defined $\mathbf{A}(y) := \mathbf{C}(y) \cdot \begin{pmatrix} \overline{\mathbf{A}}(y) & \\ & \mathbf{I}_{n-r-s} \end{pmatrix}$, where $\mathbf{C}(y) \in \mathbb{Z}_2^{k\times k}$ comes from the output $\mathsf{F}\left(k_{\mathbf{C}}, y\right)$ and

$\overline{\mathbf{A}}(y) \in \mathbb{Z}_2^{(k-(n-r-s))\times s}$ is the output of the inner PRF $\overline{\mathsf{F}}\left(\overline{k}_{\mathsf{lin}}\right)$ (which in turn comes from the inside of $\left(\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}\right)$). In the current hybrid we are going to ignore the PRFs $\mathsf{F}\left(k_{\mathbf{C}}, y\right)$ and $\overline{\mathsf{F}}\left(\overline{k}_{\mathsf{lin}}\right)$ and their generated values $\mathbf{C}(y)$, $\mathbf{d}(y)$ and $\overline{\mathbf{A}}(y)$ and instead, sample a fresh key $k_{\mathbf{A}}$, and on query y generate $\mathbf{A}(y) \leftarrow \mathsf{F}\left(k_{\mathbf{A}}, y\right)$, for $\mathbf{A}(y) \in \mathbb{Z}_2^{k\times(n-r)}$.

First, note that the following two ways to sample $\mathbf{A}(y)$, are statistically equivalent for every y: (1) The matrix $\mathbf{A}(y)$ is generated by sampling a random full-rank matrix $\mathbf{C}(y) \in \mathbb{Z}_2^{k\times k}$ and letting $\mathbf{A}(y)$ be $\mathbf{C}(y) \cdot \begin{pmatrix} \overline{\mathbf{A}}(y) & \\ & \mathbf{I}_{n-r-s} \end{pmatrix}$. (2) For every $y \in \mathbb{Z}_2^r$ just sample a full-rank matrix $\mathbf{A}(y) \in \mathbb{Z}_2^{k\times(n-r)}$. This means that when truly random bits are used for generating $\mathbf{A}(y)$ in the two cases, the distributions are statistically equivalent.

To see why the two distributions are computationally indistinguishable, a different description of the previous hybrid can be given as follows: We can consider a sampler E_0 that for every $y \in \{0,1\}^d$ samples $\mathbf{A}(y)$ according to the first algorithm, and another sampler E_1 that samples $\mathbf{A}(y)$ according to the second algorithm, and we know that for every y (and recall there are 2^r actual values of y which can appear as the output, and not 2^d) the outputs of E_0 and E_1 are statistically indistinguishable.

Since there are 2^r valid values for y, by Lemma 13, the current hybrid is computationally indistinguishable from the previous, with indistinguishability $\frac{2^r}{f(\kappa)}$.

- Hyb_3: Discarding the inner obfuscations $\left(\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}\right)$ completely, by using the security of the outer obfuscator.

The change between the current hybrid and the previous is that in the current hybrid we generate the circuits P, P^{-1}, D' without using $\left(\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}\right)$ at all. Note that this is possible, since in the previous hybrid, we moved to a circuit that did not use access to the circuits $\left(\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}\right)$ any longer during the execution of any of the three circuits P, P^{-1}, D', except from using the second permutation $\overline{\Pi}_{\mathsf{out}}$, which acts on $\{0,1\}^d$ and does not need to act from inside the inner obfuscations $\left(\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}\right)$ any more. This means that we can technically move the application of the inner permutation $\overline{\Pi}_{\mathsf{out}}$ "outside of the inner circuits $\left(\overline{\mathcal{P}}, \overline{\mathcal{P}}^{-1}\right)$" and the functionality of the circuits P, P^{-1}, D' did not change between the current and the previous hybrids, and thus, by the security of the indistinguishability obfuscator that obfuscates the three circuits, the current hybrid is computationally indistinguishable from the previous one, with indistinguishability of $\frac{1}{f(\kappa)}$.

Finalizing the reduction. Finally, observe that the distribution generated in the above Hyb_3 is exactly a sample from $\widetilde{\mathsf{Gen}}\left(1^\lambda, n, r, k, s\right)$. Also observe that the outputs of Hyb_0 and Hyb_3 are $O\left(\frac{2^r}{f(\kappa)}\right)$-computationally indistinguishable. Recall that by the lemma's assumptions, with probability ϵ, on a sample from $\widetilde{\mathsf{Gen}}\left(1^\lambda, n, r, k, s\right)$, the algorithm $\mathcal{A}$ outputs a pair (x_0, x_1) of n-bit strings such that $y_0 = y_1 := y$ and also $(\mathbf{u}_0 - \mathbf{u}_1) \notin \mathsf{ColSpan}\left(\mathbf{A}(y)^{(1)}\right)$. It follows that the probability for the same event when the input to $\mathcal{A}$ is generated by Hyb_0, is at least $\epsilon - O\left(\frac{2^r}{f(\kappa)}\right) \geq \frac{\epsilon}{2}$, which finishes our proof. □

4.3 Hardness of the Dual-free Case from Decomposable Claw-free Functions

Our main object in this section will be trapdoor claw-free functions L, which may either be exact (i.e. where every output of L has exactly 2 preimages) or approximate/noisy. We will also need the function to be "decomposable" in some sense – we will gradually explain these properties below.

Definition 33 (Trapdoor Claw-free Function)**.** *A trapdoor claw-free function scheme is given by classical algorithms* $\left(\mathsf{Gen}, L, L^{-1}\right)$ *with the following syntax.*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}\left(1^\lambda\right)$*: a probabilistic polynomial-time algorithm that gets as input the security parameter* $\lambda \in \mathbb{N}$*, and samples a public and secret key pair.*
- $y \leftarrow L\left(\mathsf{pk}, x\right)$*: a deterministic polynomial-time algorithm that gets as input the public key* pk *and an input* $x \in \{0,1\}^{\mathbf{in}_L}$ *and outputs a string* $y \in \{0,1\}^{\mathbf{out}_L}$ *for some* $\mathbf{out}_L \geq \mathbf{in}_L - 1$*.*
- $x_b \leftarrow L^{-1}\left(\mathsf{sk}, b \in \{0,1\}, y\right)$*: a deterministic polynomial-time algorithm that gets as input the public key* pk*, a string* $y \in \{0,1\}^{\mathbf{out}_L}$ *and a bit* b*. The algorithm outputs a string* $x_b \in \{0,1\}^{\mathbf{in}_L}$ *or* $\perp$*.*

The scheme satisfies the following guarantees.

- **2-to-1 mapping.** *For every public key* pk *in the support of* $\mathsf{Gen}\left(1^\lambda\right)$*, every output in the image of* $L\left(\mathsf{pk}, \cdot\right) : \{0,1\}^{\mathbf{in}_L} \to \{0,1\}^{\mathbf{out}_L}$ *has exactly either 2 or 1 preimages. In case it has 2, then these differ in their first input bit.*
- **Efficient inversion given a trapdoor.** *For every public-secret key pair* pk, sk *in the support of* $\mathsf{Gen}\left(1^\lambda\right)$ *and* $y \in \{0,1\}^{\mathbf{out}_L}$ *in the image of* $L\left(\mathsf{pk}, \cdot\right)$*, the inverse function* $L^{-1}\left(\mathsf{sk}, \cdot\right) : \left(\{0,1\} \times \{0,1\}^{\mathbf{out}_L}\right) \to \{0,1\}^{\mathbf{in}_L}$*, given input* (b, y)*, guarantees the following. In case* y *has a preimage in* $L\left(\mathsf{pk}, \cdot\right)$ *that starts with* b *then the inverse function outputs it, and otherwise it outputs* $\perp$*.*
- **Collision-resistance.** *Let* $f : \mathbb{N} \to \mathbb{N}$ *a function. We say that* L *is* $\left(f(\cdot), \frac{1}{f(\cdot)}\right)$*-collision resistant if for every quantum algorithm running in time* $f(\lambda)$*, the probability for the following experiment to succeed is bounded by* $1/f(\lambda)$*. The experiment samples* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}\left(1^\lambda\right)$*, gives* pk *to the quantum algorithm, and the algorithm needs to find a collision in* $L\left(\mathsf{pk}, \cdot\right)$*.*
- **Effective input size.** *There exists a number* $\mathbf{E}_L \in \mathbb{N}$ *which we refer to as the effective input size and it satisfies* $\mathbf{E}_L \leq \mathbf{in}_L - 1$*. Given an input* $x \in \{0,1\}^{\mathbf{in}_L}$ *we can map it to compressed input* $x^{(\mathbf{E}_L)} \in \{0,1\}^{\mathbf{E}_L}$*, by discarding the first bit of* x*, taking the following* $\mathbf{E}_L$ *bits of* x *and discarding the rest, if there are any. There is also a public mapping* $f_{\mathbf{E}_L \to \mathbf{in}_L}\left(\mathsf{pk}, \cdot\right)$ *which uses the public key* pk*, such that given the first bit of* x*, the output* y *it maps to, and its effective input* $x^{(\mathbf{E}_L)} \in \{0,1\}^{\mathbf{E}_L}$*, recovers the original* $x \in \{0,1\}^{\mathbf{in}_L}$*.*

The parallel-repetition hash $\mathcal{Q}$. Based on the above object, we construct our hash function $\mathcal{Q}$ as follows.

Construction 34 (Folding Coset Partition Function)**.** *Let L a function as in Definition 33, and let n, r natural numbers. We sample $n-r$ public keys $\mathsf{pk}_1, \cdots, \mathsf{pk}_{n-r}$ and their respective trapdoors $\mathsf{sk}_1, \cdots, \mathsf{sk}_{n-r}$. We define two functions*

$$\mathcal{Q} : \{0,1\}^{(n-r)\cdot \mathbf{in}_L} \to \left(\{0,1\}^{(n-r)\cdot \mathbf{out}_L} \times \{0,1\}^{n-r+\mathbf{E}_L} \right) \ ,$$

$$\mathcal{Q}^{-1} : \left(\{0,1\}^{(n-r)\cdot \mathbf{out}_L} \times \{0,1\}^{n-r+\mathbf{E}_L} \right) \to \{0,1\}^{(n-r)\cdot \mathbf{in}_L}$$

, as follows.

The computation $(y, \widetilde{\mathbf{w}}) \leftarrow \mathcal{Q}(\mathbf{w})$:

1. *We compute the first part of the output of $\mathcal{Q}$ by partitioning the input into $n-r$ equal-sized inputs to the instances of L:*

$$y := (y_1, \cdots, y_{n-r}) \leftarrow L_1(\mathsf{pk}_1, \mathbf{w}_1), \cdots, L_{n-r}\left(\mathsf{pk}_{n-r}, \mathbf{w}_{n-r}\right) \ .$$

2. ***Fold*** $\mathbf{w} \in \{0,1\}^{(n-r)\cdot \mathbf{in}_L}$ ***into*** $\widetilde{\mathbf{w}} \in \{0,1\}^{n-r+\mathbf{E}_L}$***:***
   (a) *Generate a coordinates vector $\mathbf{r} \in \mathbb{Z}_2^{n-r}$ such that for $i \in [n-r]$, bit i of $\mathbf{r}$ is the first bit of $\mathbf{w}_i \in \{0,1\}^{\mathbf{in}_L}$, which in turn is the input to L_i inside the computation of Q.*
   (b) *For each $\mathbf{w}_i \in \{0,1\}^{\mathbf{in}_L}$ consider $\mathbf{w}_i^{(\mathbf{E}_L)} \in \{0,1\}^{\mathbf{E}_L}$ the effective input of $\mathbf{w}_i$. Then set*

$$\widetilde{\mathbf{w}} := \left(\mathbf{r}, \sum_{i \in [n-r]} \mathbf{w}_i^{(\mathbf{E}_L)} \right) \ .$$

3. *Output $(y, \widetilde{\mathbf{w}})$.*

The computation $\mathbf{w} \leftarrow \mathcal{Q}^{-1}(y, \widetilde{\mathbf{w}})$:

1. *Parse $\widetilde{\mathbf{w}} := \left(\mathbf{r} \in \mathbb{Z}_2^{n-r}, \overline{\mathbf{w}} \in \{0,1\}^{\mathbf{E}_L}\right)$. Write $y = (y_1, \cdots, y_{n-r})$ where $y_i \in \{0,1\}^{\mathbf{out}_L}$ for every $i \in [n-r]$.*
2. *For every $j \in [n-r]$ compute $\mathbf{w}_{(j,\, y_j,\, \mathbf{r}_j)} \leftarrow L_j^{-1}(\mathsf{sk}_j, \mathbf{r}_j, y_j)$, where $\mathbf{r}_j \in \{0,1\}$ is the j-th bit of $\mathbf{r}$.*
3. *Verify match between inverted values and input: for every j compute the effective input $\mathbf{w}_{(j,\, y_j,\, \mathbf{r}_j)}^{(\mathbf{E}_L)} \in \{0,1\}^{\mathbf{E}_L}$. Check that $\sum_{j \in [n-r]} \mathbf{w}_{(j,\, y_j,\, \mathbf{r}_j)}^{(\mathbf{E}_L)} = \overline{\mathbf{w}}$ and otherwise terminate and reject.*
4. *Output $\mathbf{w} \leftarrow \left(\mathbf{w}_{(1,\, y_1,\, \mathbf{r}_1)}, \cdots, \mathbf{w}_{(n-r,\, y_{n-r},\, \mathbf{r}_{n-r})} \right)$.*

Permutation-Indistinguishability of Trapdoor Functions. We would like for L to have an additional property of decomposability. The reason we need this property is that whenever we construct the function $\mathcal{Q}$, we will be able to embed it indistinguishably within the pair of oracles $\left(\mathcal{P}, \mathcal{P}^{-1}\right)$. The property itself will be a local property we need from L, which will ask that when we put two permutable PRPs, one before the input and one after the output of L, and then obfuscate the entire circuit with iO, this will be indistinguishable from simply obfuscating the two PRPs, concatenated to each other.

Definition 35 (Permutation-Indistinguishable Trapdoor Function)**.** *Let L a trapdoor function as in Definition 33. Let $\Pi\left(k_{\mathsf{in}}, \cdot\right)$ an OP-PRP with domain $\{0,1\}^{\mathbf{in}_L}$ that's output-permutable for all $\left(2^{\mathsf{poly}(\lambda)}, \mathsf{poly}(\lambda)\right)$-decomposable permutations. Let $\Pi\left(k_{\mathsf{out}}, \cdot\right)$ an IP-PRP with domain $\{0,1\}^{\mathbf{in}_L+\mathbf{out}_L}$ that's input-permutable for all $\left(2^{\mathsf{poly}(\lambda)}, \mathsf{poly}(\lambda)\right)$-decomposable permutations. Let iO an iO scheme. We say that L is $\left(f_D, \frac{1}{f_D}\right)$-permutation-indistinguishable if the following two distributions are $\left(f_D, \frac{1}{f_D}\right)$-indistinguishable, when obfuscated using the iO:*

- *The first distribution is given by the pair f_0, f_0^{-1}. The function*

$$f_0 : \{0,1\}^{\mathbf{in}_L} \to \left(\{0,1\}^{\mathbf{in}_L+\mathbf{out}_L} \times \{0,1\}\right)$$

  applies $\Pi\left(k_{\mathsf{in}}, \cdot\right)$, then L, then pads with sufficiently-many zeros, and then applies $\Pi\left(k_{\mathsf{out}}, \cdot\right)$. It also outputs first output bit of $\Pi\left(k_{\mathsf{in}}\right)$. The function f_0^{-1} is the inverse of f_0, which can be computed using $\mathsf{pk}, \mathsf{sk}, k_{\mathsf{in}}, k_{\mathsf{out}}$.

- *The second distribution is given by the pair f_1, f_1^{-1}, which is basically the same as the previous pair, only that we discard the use of L. The function $f_1 : \{0,1\}^{\mathbf{in}_L} \to \left(\{0,1\}^{\mathbf{in}_L+\mathbf{out}_L} \times \{0,1\}\right)$ applies $\Pi\left(k_{\mathsf{in}}, \cdot\right)$ and then $\Pi\left(k_{\mathsf{out}}, \cdot\right)$. It also outputs first output bit of $\Pi\left(k_{\mathsf{in}}\right)$. The function f_1^{-1} is the inverse of f_1, which can be computed using $k_{\mathsf{in}}, k_{\mathsf{out}}$.*

Constructions of Permutation-Indistinguishable Trapdoor Claw-free Functions. We mention two constructions of L based on two different assumptions.

The LWE-based hash function L. Here, we recall an *approximate* 2-to-1 function L based on LWE which is a simplified version of the *noisy claw-free trapdoor function* developed in [BCM+18]. Let $u, v, \sigma, B, \overline{B}, q$ be parameters with the relationships described in Equation 3.

$$\begin{aligned} &\sigma = u^{\Omega(1)}, && \overline{B} = \sigma \times u^{\Omega(1)}, \\ &B \geq \overline{B} \times u^{\omega(1)}, && q \geq B \times u^{\Omega(1)}, \\ &v \geq \Omega(u \log q), && \exists \delta \in (0,1) : \frac{q}{\sigma} \leq 2^{u^{\delta}} \quad . \end{aligned} \tag{3}$$

The keys for the hash function have the form $\mathsf{pk} = (\mathbf{B}, \mathbf{c})$, where $\mathbf{B} \leftarrow \mathbb{Z}_q^{v \times u}$ and $\mathbf{c} \leftarrow \mathbf{B} \cdot \mathbf{s} + \mathbf{e} \bmod q$ where $\mathbf{s} \leftarrow \mathbb{Z}_q^u$ and the entries of $\mathbf{e} \in \mathbb{Z}_q^v$ are i.i.d. sampled from discrete Gaussians of width σ, which in turn are guaranteed (w.h.p) to have entries in $(-\overline{B}, \overline{B}]$.

We define the function $L\left(\mathsf{pk}, \cdot\right) : \mathbb{Z}_q^u \times (-B, B]^v \times \{0,1\} \to \mathbb{Z}_q^v$ as follows.

$$L\left(\left(\mathbf{B} \in \mathbb{Z}_q^{v \times u}, \mathbf{c} \in \mathbb{Z}_q^v\right), \left(\mathbf{t} \in \mathbb{Z}_q^u, \mathbf{f} \in (-B, B]^v\right), b \in \{0,1\}\right) = \mathbf{B} \cdot \mathbf{t} + \mathbf{f} + b \cdot \mathbf{c} \bmod q \quad .$$

By choosing B, q to be powers of 2, we can map the domain and range to bit-strings.

To match the above with the interface of Definition 33, note that $\mathbf{in}_L := 1+u\cdot\log(q)+v\cdot\log(B)$, $\mathbf{out}_L := v\cdot\log(q)$. The effective input size can actually be reduced to $\mathbf{E}_L = u\cdot\log(q)$, by taking it to be $\mathbf{t}$. Observe that given the public key $\left(\mathbf{B}\in\mathbb{Z}_q^{v\times u}, \mathbf{c}\in\mathbb{Z}_q^v\right)$, the effective input $\mathbf{t}$, the first bit of the original input and also the output $\mathbf{B}\cdot\mathbf{t}+\mathbf{f}+b\cdot\mathbf{c} \bmod q$, we can subtract (or not, depending on the value of the first bit of the original input) from the output the vector $\mathbf{c}$ (which is part of the public key) then also subtract $\mathbf{B}\cdot\mathbf{t}$ to obtain the noise $\mathbf{f}\in(-B,B]^v$. Now, $\mathbf{f}$ is the part of the original input which isn't included in the effective input, so we obtained the full original input. In [SZ25] (Theorem 58) it is proved that the above function, when instantiated with modulus q and LWE noise B, will be $\left(\frac{q}{B},\frac{B}{q}\right)$-permutation-indistinguishable, for a sufficiently large $f := \min\left(f_{\mathsf{iO}}, f_{\mathsf{OWF}}, f_{\mathsf{PRP}}\right)$ for the corresponding hardness functions of iO, OWFs and permutable PRPs. Also, for $u\geq\lambda$, under the $2^{\widetilde{\Omega}(\lambda)}$-hardness LWE, our LWE-based L is $2^{\widetilde{\Omega}(\lambda)}$-collision-resistant.

Construction based on Permutable PRPs. Another plausible construction uses permutable PRPs. Specifically, the function takes as input $\lambda+1$ bits and have inside two permutable PRPs $\Pi\left(k_0,\cdot\right), \Pi\left(k_1,\cdot\right)$, each acting on domain $\{0,1\}^\lambda$. The function L, for an input $\left(b\in\{0,1\}, x\in\{0,1\}^\lambda\right)$, outputs $\Pi\left(k_b, x\right)$. We do not know how to prove that this construction is collision-resistant. We note that while the construction uses only an indistinguishability obfuscator and PRPs, and while there are impossibility results for getting collision-resistance from iO and OWFs alone, may still get collision resistance as long as *there exists* a reduction and (possibly additional security assumptions) that works. This was the same situation, for example, for the distribution $\left(\mathcal{P},\mathcal{P}^{-1}\right)$ when we use the LWE-based function L, but only later in the hybrids of the security proof.

Collision-resistance of the dual-free distribution $\left(\mathcal{P},\mathcal{P}^{-1}\right)$. Given the above properties of our base function L and the structure of $\mathcal{Q}$ given L, we are now ready to prove the collision-resistance of our functions $\mathcal{P}$, $\mathcal{P}^{-1}$.

Theorem 36 (Collision Resistance of $\left(\mathcal{P},\mathcal{P}^{-1}\right)$)**.** *For our primitives in Construction 30, let the* iO *scheme be* $\left(f_{\mathsf{iO}}\left(\cdot\right), \frac{1}{f_{\mathsf{iO}}(\cdot)}\right)$*-secure, the puncturable PRF be* $\left(f_{\mathsf{PRF}}\left(\cdot\right), \frac{1}{f_{\mathsf{PRF}}(\cdot)}\right)$*-secure, the permutable PRP be* $\left(f_{\mathsf{PRP}}\left(\cdot\right), \frac{1}{f_{\mathsf{PRP}}(\cdot)}\right)$*-secure and let* $f := \min\left(f_{\mathsf{iO}}, f_{\mathsf{PRF}}, f_{\mathsf{PRP}}\right)$.

Let $n,r,k\in\mathbb{N}$ *such that* $r<n$ *and also* $\frac{n}{n-r}$ *is an integer (which implies that* $\frac{r}{n-r}$ *is also an integer). Let* $n/(n-r) := \mathbf{in}_L$ *and suppose that* $n-r+\mathbf{E}_L\leq k$ *and that* L *is a trapdoor claw-free function that's* $\left(f_L\left(\mathbf{E}_L\right), \frac{1}{f_L(\mathbf{E}_L)}\right)$*-collision-resistant and also* $\left(f_D\left(\mathbf{E}_L\right), \frac{1}{f_D(\mathbf{E}_L)}\right)$*-permutation-indistinguishable (as in Definition 35).*

Then, getting $\left(\mathcal{P},\mathcal{P}^{-1}\right)$ *sampled from* $\left(\mathcal{P},\mathcal{P}^{-1},\mathcal{D}\right)\leftarrow\widetilde{\mathsf{Gen}}\left(1^\lambda,n,r,k,0\right)$*, and finding a collision in* Hash *(the function derived from* $\mathcal{P}$*), is* $\left(\min\left(f_L\left(\mathbf{E}_L\right), f_D\left(\mathbf{E}_L\right)\right), \frac{1}{f_L(\mathbf{E}_L)}+\frac{2^n}{f(\kappa)}+\frac{n-r}{f_D(\mathbf{E}_L)}\right)$*-hard.*

Proof. Suppose there exists an adversary $\mathcal{A}$ which, given $\left(\mathcal{P},\mathcal{P}^{-1}\right)$ as sampled from $\left(\mathcal{P},\mathcal{P}^{-1},\mathcal{D}\right)\leftarrow\widetilde{\mathsf{Gen}}\left(1^\lambda,n,r,k,0\right)$ (recall that $\mathcal{A}$ gets the sampled circuits without the dual $\mathcal{D}$), finds a collision in the associated function $\mathsf{Hash}(x) := \Pi^{-1}\left(k_{\mathsf{out}},\ H(x)||0^{d-r}\right)$, with non-negligible probability ϵ, where recall that $H(\cdot)$ applies $\Pi\left(k_{\mathsf{in}},\cdot\right)$ and outputs the first r bits of the output. We will describe an adversary $\mathcal{B}$ that violates the security of L, using the adversary $\mathcal{A}$.

Consider the function $\mathcal{Q}$ from Construction 34. Now, given our definition of the hash functions L and $\mathcal{Q}$, we are ready to describe our reduction.

The reduction $\mathcal{B}$ from collision finding in $\mathcal{Q}$ to collision finding in Hash. Given $\mathsf{pk} = \left(\mathsf{pk}_1, \cdots, \mathsf{pk}_{n-r}\right)$ a public key for $\mathcal{Q}$ (where pk_i is the i.i.d. sampled public key for L), and a trapdoor td which consists of the secret keys of the corresponding instances of L (and is used to compute the inverse $\mathcal{Q}^{-1}$) the reduction $\mathcal{B}$ samples permutable PRP keys k_{in}, k_{out} and a puncturable PRF key k'_{lin}. Denote by $\ell := (n-r) \cdot \mathbf{out}_L$ the output size of $\mathcal{Q}$, and we next define the functions P, P^{-1} which we then obfuscate to get the circuits $\mathcal{P}$, $\mathcal{P}^{-1}$, which we will feed to $\mathcal{A}$.

- $P\left(x \in \{0,1\}^n\right)$:
  1. $\mathbf{w} \leftarrow \Pi\left(k_{\mathsf{in}}, x\right)$,
  2. $\left(\mathbf{a}, \widetilde{\mathbf{w}}\right) \leftarrow \mathcal{Q}\left(\mathsf{pk}, \mathbf{w}\right)$,
  3. $y \leftarrow \Pi^{-1}\left(k_{\mathsf{out}}, \left(\mathbf{a} || 0^{d-\ell}\right)\right)$,
  4. $\left(\mathbf{C}_y \in \mathbb{Z}_2^{k \times (n-r+\mathbf{E}_L)}, \mathbf{d}_y \in \mathbb{Z}_2^k\right) \leftarrow \mathsf{F}\left(k'_{\mathsf{lin}}, y\right)$,
  5. $\mathbf{u} \leftarrow \mathbf{C}_y \cdot \widetilde{\mathbf{w}} + \mathbf{d}_y$,
  6. Output $\left(y \in \{0,1\}^d, \mathbf{u} \in \mathbb{Z}_2^k\right)$.
- $P^{-1}\left(y \in \{0,1\}^d, \mathbf{u} \in \mathbb{Z}_2^k\right)$:
  1. $x \leftarrow \begin{cases} \Pi^{-1}\left(k_{\mathsf{in}}, \mathcal{Q}^{-1}\left(\mathsf{td}, y, \widetilde{\mathbf{w}}\right)\right) & \exists \widetilde{\mathbf{w}} \in \mathbb{Z}_2^{n-r+\mathbf{E}_L} \text{ such that } \mathbf{C}_y \cdot \widetilde{\mathbf{w}} + \mathbf{d}_y = \mathbf{u} \\ \bot & \text{if no such } \mathbf{w} \text{ exists} \end{cases}$
  2. Output $\begin{cases} x & \text{if } x \neq \bot \text{ and } y = \Pi^{-1}\left(k_{\mathsf{out}}, \left(\mathbf{a} || 0^{d-\ell}\right)\right), \text{ for } \mathbf{a} \leftarrow \mathcal{Q}\left(\mathsf{pk}, \mathbf{w}\right) \\ \bot & \text{otherwise} \end{cases}$

$\mathcal{B}$ obfuscates the two circuits to get $\mathcal{P}$, $\mathcal{P}^{-1}$ and executes $\left(x_0, x_1\right) \leftarrow \mathcal{A}\left(\mathcal{P}, \mathcal{P}^{-1}\right)$. The output of $\mathcal{B}$ is $\left(\mathbf{w}_0 := \Pi\left(k_{\mathsf{in}}, x_0\right), \mathbf{w}_1 := \Pi\left(k_{\mathsf{in}}, x_1\right)\right)$ as a collision in $\mathcal{Q}$. Observe that any collision in the simulated $\left(\mathcal{P}, \mathcal{P}^{-1}\right)$ implies a collision in $\mathcal{Q}$. Our proof has two parts. First we show that the above distribution generated by the reduction is collision resistant. In the second part we show that it is indistinguishable from the original $\mathcal{P}, \mathcal{P}^{-1}$.

Lemma 37 (Collision-resistance of $\mathcal{Q}$, $\mathcal{Q}^{-1}$ under obfuscation)**.** *Assume that L is $\left(f(\mathbf{E}_L), \frac{1}{f(\mathbf{E}_L)}\right)$-collision resistant. Then, the above distribution generated by $\mathcal{B}$ is $\left(f(\mathbf{E}_L) - \mathsf{poly}(\lambda), \frac{2 \cdot (n-r)}{f(\mathbf{E}_L)}\right)$-collision resistant.*

Proof. The key to our proof is the ability to choose any single index $i^* \in [n-r]$ in the parallel repetition of L inside $\mathcal{Q}$, and simulate both functionalities $\mathcal{Q}$, $\mathcal{Q}^{-1}$ with a logically equivalent circuit, without knowing the secret key of instance i^*.

Formally, assume there is a quantum algorithm $\mathcal{A}$ that given $\widehat{\mathcal{Q}}$, $\widehat{\mathcal{Q}}^{-1}$ obfuscations of the functions, finds a collision in $\mathcal{Q}$ with probability ϵ. We describe an algorithm $\mathcal{B}$ that will attack the collision resistance of L. First, note that since $\mathcal{A}$ finds a collision, if we partition the n-bit input to $\mathcal{Q}$ into $n-r$ packets of $n/(n-r)$ bits each, then there exists a packet $i^* \in [n-r]$ such that we get a collision for it with probability at least $\epsilon/(n-r)$.

$\mathcal{B}$ gets pk^* a public key for L. It samples $n-r-1$ public-secret key pairs $\left(\mathsf{pk}_1, \mathsf{sk}_1, \cdots, \mathsf{pk}_{n-r}, \mathsf{sk}_{n-r}\right)$ of claw-free functions $L_1, \cdots, L_{n-r-1}$. We next denote $L'_1\left(\mathsf{pk}'_1, \cdot\right), \cdots, L'_{n-r}\left(\mathsf{pk}'_{n-r}, \cdot\right)$, where instance number i^* is given by $L\left(\mathsf{pk}^*, \cdot\right)$, that is $\mathsf{pk}'_{i^*} := \mathsf{pk}^*$.

Next, consider the functions

$$\mathcal{Q}_{sim} : \{0,1\}^n \to \left(\{0,1\}^{(n-r)\cdot \mathbf{out}_L} \times \{0,1\}^{n-r+\mathbf{E}_L}\right) \ ,$$

$$\mathcal{Q}_{sim}^{-1} : \left(\{0,1\}^{(n-r)\cdot \mathbf{out}_L} \times \{0,1\}^{n-r+\mathbf{E}_L}\right) \to \{0,1\}^n \ .$$

, defined as follows.

The computation $(y, \widetilde{\mathbf{w}}) \leftarrow \mathcal{Q}_{sim}(\mathbf{w})$:

1. Identical to the original $\mathcal{Q}$ computes its output, using the functions $\left(L'_j\right)_{j\in[n-r]}$. Note that $\mathcal{Q}$ only uses the public keys so we have the information we need in $\mathcal{Q}_{sim}$.

The computation $\mathbf{w} \leftarrow \mathcal{Q}_{sim}^{-1}(y, \widetilde{\mathbf{w}})$:

1. Parse $\widetilde{\mathbf{w}} := \left(\mathbf{r} \in \mathbb{Z}_2^{n-r}, \overline{\mathbf{w}} \in \{0,1\}^{\mathbf{E}_L}\right)$. Write $y = (y_1, \cdots, y_{n-r})$ where $y_i \in \{0,1\}^{\mathbf{out}_L}$ for every $i \in [n-r]$.
2. For every $j \in [n-r] \setminus \{i^*\}$ compute $\mathbf{w}_{(j,\, y_j,\, \mathbf{r}_j)} \leftarrow L_j^{-1}(\mathsf{sk}_j, \mathbf{r}_j, y_j)$, where $\mathbf{r}_j \in \{0,1\}$ is the j-th bit of $\mathbf{r}$.
3. For every $j \in [n-r] \setminus \{i^*\}$ let $\left(\mathbf{w}_{(j,\, y_j,\, \mathbf{r}_j)}\right)_{-1} \in \{0,1\}^{\mathbf{E}_L}$ which is derived by discarding the first bit of $\mathbf{w}_{(j,\, y_j,\, \mathbf{r}_j)}$ and then taking the following $\mathbf{E}_L$ bits (and discarding the rest).
4. Subtract to get
$$\mathbf{w}^* \leftarrow \overline{\mathbf{w}} - \sum_{j\in[n-r]\setminus\{i^*\}} \mathbf{w}_{(j,\, y_j,\, \mathbf{r}_j)} \ .$$
5. Obtain full input from effective input:
$$(\mathbf{w}^*, y_{i^*}) \mapsto_{\mathsf{pk}_{i^*}} \mathbf{w}_{(i^*,\, y_{i^*},\, \mathbf{r}_{i^*})} \ .$$
6. Verify that $y_{i^*} = L\left(\mathsf{pk}_*, \mathbf{w}_{(i^*,\, y_{i^*},\, \mathbf{r}_{i^*})}\right)$ and otherwise terminate and reject.
7. Output $\mathbf{w} \leftarrow \left(\mathbf{w}_{(1,\, y_1,\, \mathbf{r}_1)}, \cdots, \mathbf{w}_{(n-r,\, y_{n-r},\, \mathbf{r}_{n-r})}\right)$.

Finalizing the reduction. One can observe that the functionality of $\mathcal{Q}$ and $\mathcal{Q}_{sim}$ is identical, and the functionality of $\mathcal{Q}^{-1}$ is also identical to that of $\mathcal{Q}_{sim}^{-1}$, for every choice of public keys $\mathsf{pk}_1, \cdots, \mathsf{pk}_{n-r-1}$ and also pk^*. In particular, the choice of i^* did not change the functionality at all – only the inner workings of the functions changed, but not the . This means that under obfuscation, we have indistinguishability by the security of the iO. Now, since we get collisions on packet $i^* \in [n-r]$ with probability $\epsilon/(n-r)$, this means our collisions will be on the packet that corresponds to the input for $L(\mathsf{pk}^*, \cdot)$. This finishes our proof. □

We next prove that the distribution which is simulated by the reduction is indistinguishable from the riginal dual-free, which will prove that the dual-free distribution is collision-resistant. Let $\epsilon_{\mathcal{B}}$ the probability that $\mathcal{B}$ outputs a collision in $\mathcal{Q}$. We next define a sequence of hybrid experiments. Each hybrid defines a computational process, an output of the process and a predicate computed on the process output. The predicate defines whether the (hybrid) process execution was successful or not.

- Hyb_0: The original execution of the reduction $\mathcal{B}$.

Here, we get the distribution generated by the reduction and we execute $(\mathbf{w}_0, \mathbf{w}_1) \leftarrow \mathcal{B}\left(\widetilde{\mathcal{Q}}, \widetilde{\mathcal{Q}}^{-1}\right)$. The output of this hybrid is $(\mathbf{w}_0, \mathbf{w}_1)$ and the process is defined as successful if the pair constitutes a collision in $\mathcal{Q}(\mathsf{pk}, \cdot)$. By definition, the success probability of this hybrid is $\epsilon_0 := \epsilon_{\mathcal{B}}$.

- Hyb_1: Generating $\mathbf{u}$ from a coordinates vector $\mathbf{z}$ and the coset $\left(\overline{\mathbf{A}}_{\mathbf{a}}, \overline{\mathbf{b}}_{\mathbf{a}}\right)$ instead of the preimage $\mathbf{w}$, by using the trapdoor of $\mathcal{Q}$ and the security of the iO. Also, changing back to asking for collisions in the original $\mathcal{P}$.

Here, the public key pk for $\mathcal{Q}$ is sampled together with a trapdoor td. The change we make to the following hybrid is this: In the previous hybrid, we took $\mathbf{a}$ and computed $y \in \{0,1\}^d$, then computed $(\mathbf{C}_y, \mathbf{d}_y) \leftarrow \mathsf{F}(k'_{\mathsf{lin}}, y)$ and set $\mathbf{u} \leftarrow \mathbf{C}_y \cdot \widetilde{\mathbf{w}} + \mathbf{d}_y$. Here, we compute $(\mathbf{C}_y, \mathbf{d}_y)$ all the same, but instead of using $\widetilde{\mathbf{w}}$ to compute $\mathbf{u}$, we use (1) the coordinates vector $\mathbf{z}$ of $\mathbf{w}$ and the fact that it is publicly computable, (2) the output y and (3) the trapdoor td.

Specifically, recall that given an image $\mathbf{a} \in \{0,1\}^{(n-r)\cdot \mathbf{out}_L}$ of $\mathcal{Q}(\mathsf{pk}, \cdot)$ and the trapdoor td, we can efficiently compute the coset $\left(\overline{\mathbf{A}}_{\mathbf{a}}, \overline{\mathbf{b}}_{\mathbf{a}}\right)$ and also, given an input $\mathbf{w}$ we can compute $\mathbf{z}$ the coordinates vector of $\mathbf{w}$ with respect to the coset of the output $\mathbf{a}$ (publicly with no trapdoor). Given y we use the key k'_{lin} to compute $\mathbf{a}$ and then use td to compute $\left(\overline{\mathbf{A}}_{\mathbf{a}}, \overline{\mathbf{b}}_{\mathbf{a}}\right)$. Now, we take $\mathbf{z}$ the last $n-r$ bits of $\mathbf{w}$ and set $\mathbf{u} \leftarrow \mathbf{A}_y \cdot \mathbf{z} + \mathbf{d}_y$ for $\mathbf{A}_y := \mathbf{C}_y \cdot \overline{\mathbf{A}}_{\mathbf{a}}$, $\mathbf{d}_y := \mathbf{d}_y + \mathbf{C}_y \cdot \overline{\mathbf{b}}_{\mathbf{a}}$.

By the correctness of the trapdoor, the functionality of the circuits did not change. Thus, the circuits given to $\mathcal{A}$ in Hyb_0 are computationally indistinguishable by the security of the iO that obfuscates the circuits P, P^{-1}. It follows in particular that the success probability of the current process is $:= \epsilon_1$ such that $\epsilon_0 \geq \epsilon_1 - \frac{1}{f(\kappa)}$.

Another change that we make to this hybrid the definition of successful execution: Instead of asking for collisions in $\mathcal{Q}$, we ask for collisions in the original $\mathcal{P}$. Since we move back and forth between collisions in $\mathcal{Q}$ and $\mathcal{P}$ using a permutation, any collision in $\mathcal{Q}$ can be translated into a collision in $\mathcal{P}$. Formally the reduction $\mathcal{B}$ changes minimally: After executing $\mathcal{A}$ and obtaining (x_0, x_1) we do not apply $\Pi(k_{\mathsf{in}}, \cdot)$ in the end to obtain $\mathbf{w}_0$, $\mathbf{w}_1$. The success probability still satisfies $\epsilon_0 \geq \epsilon_1 - \frac{1}{f(\kappa)}$.

- Hyb_2: Discarding the trapdoor td and computing the coset $(\mathbf{A}_y, \mathbf{b}_y)$ as a function of y alone, using an obfuscated puncturable PRF.

The change we will make to the following hybrid is to the circuit that samples the coset $(\mathbf{A}_y, \mathbf{b}_y)$. Specifically, instead of sampling the coset through the process from previous hybrid (setting $\mathbf{A}_y := \mathbf{C}_y \cdot \overline{\mathbf{A}}_{\mathbf{a}}$, $\mathbf{d}_y := \mathbf{d}_y + \mathbf{C}_y \cdot \overline{\mathbf{b}}_{\mathbf{a}}$ for the pair $\left(\overline{\mathbf{A}}_{\mathbf{a}}, \overline{\mathbf{b}}_{\mathbf{b}}\right)$ arising from the trapdoor td and the pseudorandomly generated $(\mathbf{C}_y, \mathbf{d}_y) \leftarrow \mathsf{F}(k'_{\mathsf{lin}}, y)$), we just sample a fresh puncturable PRF key k_{lin} and sample the coset description from scratch $\left(\mathbf{A}_y \in \mathbb{Z}_2^{k \times (n-r)}, \mathbf{b}_y \in \mathbb{Z}_2^k\right) \leftarrow \mathsf{F}(k_{\mathsf{lin}}, y)$.

Note that in the first distribution (arising from the sampling process of Hyb_1), the matrix $\overline{\mathbf{A}}_{\mathbf{a}}$ is full rank and thus for a truly random $(\mathbf{C}_y, \mathbf{d}_y)$, the pair $(\mathbf{A}_y, \mathbf{b}_y)$ is a truly random coset. To see why the two distributions are computationally indistinguishable, a different description of the previous hybrid can be given as follows: We can consider a sampler E_0 that for every $y \in \{0,1\}^d$ samples $\mathbf{A}(y)$ according to the first algorithm, and another sampler E_1 that samples $\mathbf{A}(y)$ according to the second algorithm, and we know that for every y (and recall there are at most 2^n actual values of y which can appear as the output, and not 2^d) the outputs of E_0 and E_1 are statistically equivalent.

Since there are $\leq 2^n$ valid values for y, by Lemma 13, the current hybrid is $\frac{2^n}{f(\kappa)}$-indistinguishable from the previous. It follows in particular that the success probability of the current process is $:= \epsilon_2$ such that $\epsilon_1 \geq \epsilon_2 - \frac{2^n}{f(\kappa)}$.

- Hyb_3: The original distribution of the obfuscated dual-free construction $\left(P, P^{-1}\right)$, using the permutation-indistinguishability of L (which implies the permutation-indistinguishability of $\mathcal{Q}$).

Note that once we moved to the previous hybrid, the program that's obfuscated applies the original circuit P from the Construction 30, only applying $\mathcal{Q}$ in the middle, between the two permutations $\Pi\left(k_{\mathsf{in}}, \cdot\right)$ and $\Pi\left(k_{\mathsf{out}}, \cdot\right)$.

In the current last hybrid we simply output $\left(\mathcal{P}, \mathcal{P}^{-1}\right)$ for $\left(\mathcal{P}, \mathcal{P}^{-1}, \mathcal{D}\right) \leftarrow \widetilde{\mathsf{Gen}}\left(1^\lambda, n, r, k, 0\right)$. That is, we disregard the execution of $\mathcal{Q}$ in the middle of the permutations. Here, we use the permutation-indistinguishability of L (as per Definition 35). This in turn implies the permutation-indistinguishability of $\mathcal{Q}$, and finally, by the security of the two permutable PRPs (in the input and output of P, P^{-1}) and also the security of the iO that obfuscates the circuits P, P^{-1}, the output in this hybrid is $\left(\frac{n-r}{f_D(\mathbf{E}_L)} + \frac{n-r}{f(\kappa)}\right)$-indistinguishable from the previous. This means in particular that the success probability in the current hybrid is $:= \epsilon_3$ such that $\epsilon_3 \geq \epsilon_2 - \left(\frac{n-r}{f_D(\mathbf{E}_L)} + \frac{n-r}{f(\kappa)}\right)$. This finishes our proof. □

References

[Aar09] Scott Aaronson. Quantum copy-protection and quantum money. In *Proceedings of the 2009 24th Annual IEEE Conference on Computational Complexity*, CCC '09, page 229–242, USA, 2009. IEEE Computer Society.

[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.

[AGQY22] Prabhanjan Ananth, Aditya Gulati, Luowen Qian, and Henry Yuen. Pseudorandom (function-like) quantum state generators: New definitions and applications. In *Theory of Cryptography Conference*, pages 237–265. Springer, 2022.

[AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 57–74, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany.

[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.

[AS15] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 191–209, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press.

[BBV24] James Bartusek, Zvika Brakerski, and Vinod Vaikuntanathan. Quantum state obfuscation from classical oracles. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *56th Annual ACM Symposium on Theory of Computing*, pages 1009–1017, Vancouver, BC, Canada, June 24–28, 2024. ACM Press.

[BCM+18] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In Mikkel Thorup, editor, *59th Annual Symposium on Foundations of Computer Science*, pages 320–331, Paris, France, October 7–9, 2018. IEEE Computer Society Press.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *Theory of cryptography conference*, pages 52–73. Springer, 2014.

[BDS23] Shalev Ben-David and Or Sattath. Quantum Tokens for Digital Signatures. *Quantum*, 7:901, January 2023.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer Berlin Heidelberg, Germany.

[BGMS25] James Bartusek, Aparna Gupte, Saachi Mutreja, and Omri Shmueli. Classical obfuscation of quantum circuits via publicly-verifiable qfhe. *arXiv preprint arXiv:2510.08400*, 2025.

[BGS13] Anne Broadbent, Gus Gutoski, and Douglas Stebila. Quantum one-time programs - (extended abstract). In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 344–360, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany.

[BKNY23] James Bartusek, Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Obfuscation of pseudo-deterministic quantum circuits. In Barna Saha and Rocco A. Servedio, editors, *55th Annual ACM Symposium on Theory of Computing*, pages 1567–1578, Orlando, FL, USA, June 20–23, 2023. ACM Press.

[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. Unclonable cryptography with unbounded collusions and impossibility of hyperefficient shadow tomography. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024: 22nd Theory of Cryptography Conference, Part III*, volume 15366 of *Lecture Notes in Computer Science*, pages 225–256, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland.

[ÇGS25] Alper Çakan, Vipul Goyal, and Omri Shmueli. Public-key quantum fire and key-fire from classical oracles. QCRYPT, 2025.

[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.

[CS20] Andrea Coladangelo and Or Sattath. A quantum money solution to the blockchain scalability problem. *Quantum*, 4:297, 2020.

[Dra23] Justin Drake. One-shot signatures. talk given at the programmable cryptography conference progcrypto, 2023. https://www.youtube.com/watch?v=VmqkH3NPG_s.

[DS23] Marcel Dall'Agnol and Nicholas Spooner. On the Necessity of Collapsing for Post-Quantum and Quantum Commitments. In Omar Fawzi and Michael Walter, editors, *18th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2023)*, volume 266 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:23, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[GLR+25] Aparna Gupte, Jiahui Liu, Justin Raizes, Bhaskar Roberts, and Vinod Vaikuntanathan. Quantum one-time programs, revisited. In Michal Koucký and Nikhil Bansal, editors, *57th Annual ACM Symposium on Theory of Computing*, pages 213–221, Prague, Czechia, June 23–27, 2025. ACM Press.

[GZ20] Marios Georgiou and Mark Zhandry. Unclonable decryption keys. Cryptology ePrint Archive, Report 2020/877, 2020.

[HSVZ26] Andrew Huang, Omri Shmueli, Vinod Vaikuntanathan, and Mark Zhandry. Uncloneable cryptography in linear quantum memory. *Cryptology ePrint Archive*, 2026.

[HT25] Mi-Ying Huang and Er-Cheng Tang. Obfuscation of unitary quantum programs. *arXiv preprint arXiv:2507.11970*, 2025.

[HV25] Andrew Huang and Vinod Vaikuntanathan. A simple and efficient one-shot signature scheme, 2025.

[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.

[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).

[Pin65] M. S. Pinsker. On the complexity of decoding. *Problemy Peredachi Informatsii*, 1(1):113–116, 1965. English translation in *Problems of Information Transmission*, vol. 1(1), 1965.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 187–196, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

[Sat22] Or Sattath. Quantum prudent contracts with applications to bitcoin, 2022.

[Shm22a] Omri Shmueli. Public-key quantum money with a classical bank. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 790–803, 2022.

[Shm22b] 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.

[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.

[Unr16] Dominique Unruh. Computationally binding quantum commitments. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 497–527. Springer, 2016.

[WZ24] Brent Waters and Mark Zhandry. Adaptive security in SNARGs via iO and lossy functions. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part X*, volume 14929 of *Lecture Notes in Computer Science*, pages 72–104, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[Zha15] Mark Zhandry. A note on the quantum collision and set equality problems. *Quantum Info. Comput.*, 15(7–8):557–567, May 2015.

[Zha19] Mark Zhandry. Quantum lightning never strikes the same state twice. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 408–438, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland.

[Zha21a] Mark Zhandry. How to construct quantum random functions. *Journal of the ACM (JACM)*, 68(5):1–43, 2021.

[Zha21b] Mark Zhandry. Quantum lightning never strikes the same state twice. or: quantum money from cryptographic assumptions. *Journal of Cryptology*, 34(1):6, 2021.