Delegation of Computation without Rejection Problem from Designated Verifier CS-Proofs*

Shafi Goldwasser† Huijia Lin‡ Aviad Rubinstein §

Abstract

We present a designated verifier CS proof system for polynomial time computations. The proof system can only be verified by a designated verifier: one who has published a public-key for which it knows a matching secret key unknown to the prover. Whereas Micali's CS proofs require the existence of random oracles, we can base soundness on computational assumptions: the existence of leveled fully homomorphic encryption (FHE) schemes, the DDH assumption and a new knowledge of exponent assumption.

Using our designated verifier CS proof system, we construct two schemes for delegating (polynomial-time) computation. In such schemes, a delegator outsources the computation of a function F on input x to a polynomial time worker, who computes the output $y = F(x)$ and proves to the delegator the correctness of the output.

Let T be the complexity of computing F on inputs of length $n = |x|$ and let k be a security parameter. Our first scheme calls for an one-time off-line stage where the delegator sends a message to the worker, and a non-interactive on-line stage where the worker sends the output together with a certificate of correctness to the prover per input x. The total computational complexity of the delegator during off-line and on-line stages is $\text{poly}(k, n, \log T)$. Compared with previous constructions by Gennaro-Gentry-Parno and Chung-Kalai-Vadhan [GGP10, CKV10] based on FHE, their on-line stage consists of two messages and their off-line stage has (delegator's) complexity of $\text{poly}(k, n, T)$. Thus, they achieve delegator complexity $\text{poly}(k, n, \log T)$ only in an amortized sense. Compared with the construction of [GKR08] based on poly-log PIR, our first construction can handle any polynomial-time computable F rather than being restricted to $\mathcal{NC}$ computable F. Our second scheme requires no off-line stage and has a two-message "on-line" stage with complexity of $\text{poly}(k, n, \log T)$. Most importantly, it achieves *robust soundness* that guarantees that it is infeasible for a cheating worker to convince the delegator of an invalid output even if the worker learns whether the delegator accepts or rejects previous outputs and proofs. Previously the only two-round protocol that achieves robust soundness under a computational assumption appeared in [GKR08] and is restricted to only $\mathcal{NC}$ computations.

*This material is based on research sponsored in part by NSF Contract CCF-1018064, NSF Contract CCF-0729011, and the Air Force Research Laboratory under agreement number FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

†MIT, E-Mail: shafi@theory.csail.mit.edu.

‡MIT, E-Mail: huijia@csail.mit.edu.

§Tel Aviv University aviadrub@mail.tau.ac.il

1 Introduction

Efficient proof verification lies at the heart of complexity theory and the definition of the class $\mathcal{NP}$. Classically, this was captured by the idea of having a deterministic polynomial time (in the size of the statement to be proved) verification procedure which receives a proof—a polynomial size certificate—and checks for correctness. Extending classical proof systems, interactive proof systems (IP) provided a model in which the polynomial time verification procedure—the *verifier* algorithm—is randomized and can interact with an all powerful *prover* algorithm which may employ an adaptive strategy in answering the verifier's messages. This strenghening enables efficient verification of all $\mathcal{PSPACE}$ computation. In another notion of probabilistically-checkable proof systems (PCPs), the verifier is similarily randomized, but the prover is restricted to utilize a non-adaptive strategy and the soundness of the proof is guaranteed only with respect to such provers. As a result, PCPs can be used to prove any computation in $\mathcal{NEXP}$. In yet a different line of work, interactive argument systems and CS proof systems restricted the cheating provers in another way: they consider only cheating provers that run in time bounded by some function in the complexity of the statement to be proved. By relaxing the soundness requirement to hold only against computationally bounded provers, CS proofs can handle computation up to $\mathcal{NEXP}$. Overall, the works on efficient verifiability throughout the 70's and 80's—from the works on $\mathcal{NP}$ to interactive proofs, PCPs to arguments and CS proofs—dedicated the bulk of their attention *"upwards"*, trying to achieve efficient (polynomial-time) verifiability for larger and larger classes of intractable (super polynomial-time) languages.

In contrast, more recent developments starting with the work of Goldwasser, Kalai, and Rothblum [GKR08] have focused on "looking downwards", trying to obtain *extremely efficient* (say, linear time) proof verification for *tractable* languages. This line of research is motivated by real-world applications, in particular, the application of outsourcing computation: In today's network, there are computational devices of vastly different computational capabilities interacting with each other over the network. Some of these devices are computationally weak due to various resource constraints. To potentially allow a weak device to support a larger range of applications, including those that are beyond its reach computationally, a natural solution is to have a weak device—called the *delegator* in this case—to delegate computations that are too expensive to a more powerful device—called the *worker*—that is connected with the weak device via a network. Then the delegation problem is: how can a delegator be assured using its meager powers that the worker performed the computation correctly?

The connection between the delegation problem and efficient proof verification is clear. View the delegator as a verifier and the worker as a prover that computes the result of a (pre-specified) polynomial time algorithm M on some input x and prove to the verifier that $M(x) = y$. The input x is thought of as dictated to the prover by an outside source, or by the verifier itself. [1] The challenge of the delegation problem is to design a proof system which achieves "two-fold" efficiency: (1) relative efficiency of the prover, that is, the prover (i.e., the worker) can convince the verifier (i.e., the delegator) that $M(x) = y$ *without doing much more work* than evaluating M on x and (2) absolute efficiency of the verifier, that is, the verification requires *significantly less work* than evaluating M on x.

More formally, a delegation system consists of a pair of polynomial time interactive Turing machines (D, W) which, on input a polynomial time Turing machine M , an input x, and a security parameter 1^k, satisfy the following properties:

- The computational complexity of the Workder/Prover W is polynomial in the computational complexity of the algorithm M.

[1] The honest prover is restricted to be a polynomial time algorithm without having access to an auxiliary oracle or auxiliary input to help in the proving process, as was the case in argument systems in the cryptographic area.

- The computational complexity of the Delegator/Verifier D is polynomial in the size of the input x and poly-logarithmic in the computational complexity of M.
- The communication complexity of the system is poly-logarithmic in the size of the input x and poly-logarithmic in the computational complexity of M.
- *Completeness:* If y is the valid output of $M(x)$, then the Worker W convinces the Delegator D of the correctness of y with probability close to 1.
- *Soundness:* No cheating worker W' can convince the (honest) verifier to accept an incorrect output $y' \neq M(x)$ with non-negligible probability in the security parameter k.

Some remarks are in order. First, in the above formalization, we phrased the delegation problem as receiving the input x as an input provided from the outside. In previous works, the input x is often provided in a first message from verifer to prover, which allowed solutions in which the communication complexity was linear in n beyond the need to communicate x. We thus purposely stated x as an outside input to properly count the communication complexity requirement of protocols. Obviously, if an application calls for the verifier to send the input x to the prover such a message can be added. See Remark 2 in Section 5.2 for more detials. Second, in some settings it may be required to keep x secret from the worker. We consider this a fundamentally different problem which we do not address in this work. Finally, we allow the verifier to run in time polynomial in the length of the input x. One can actually remove this restriction and allow the verifier to run in time poly-logarithmic in the length of the input if it has access to an encoding of the input using an error correcting code. See Remark 3 in Section 6.1 for more details.

Below for convenience, we will use the name delegator and verifier, as well as worker and prover, interchangeably. We will also assume that the running time t_x of the delegated algorithm M on input x is bigger than the length of the input x and that of the output $y = M(x)$.

1.1 Our Results

The delegation problem is the focus of our work. To this end, we propose two solutions.

1.1.1 Solution 1: Non-Interactive Delegation—Designated Verifier CS-Proofs

The notion of Computationally-Sound (CS) proofs of [Mic00] when restricted to the case of polynomiall time computation is indeed the holy-grail of the delegation problem.

Let k be a security parameter and n is the size of the input x. CS-proofs provide a *non-interactive* delegation system where to prove that $y = M(x)$, the worker in time $\text{poly}(k, t_x)$ can generate a certificate σ of the correctness of y that has size poly-logarithmic in the running time t_x of M on input x. The verifier on input x, after receiving the certificate σ, verifies the correctness of the output y in time $\text{poly}(k, |x|, |\sigma|) = \text{poly}(k, n, \log t_x)$ (actually, the verifier runs in time quasi-linear in the input length $\tilde{O}(n) \cdot \text{poly}(k, \log t_x)$). Additionally, the prover generates the certificate in time $\text{poly}(k, t_x)$. We note that the CS proof provides a delegation scheme that achieves *instance-based* , complexity that requires the running time of the worker, who is delegated the computation of M on an input x, to depend on the running time t_x of M on that particular instance x (instead of the worst-case complexity of M). A CS-proof is publically verifiable by any algorithm with access to x, M and σ, whereas no such requirement is enforced in a delegation scheme.

Micali's notion of CS proofs [Mic00] achieves such wondrous non-interactive delegation system, but, with a heavy price: The construction of CS proofs needs to revert to the *random oracle model*. In essence, the construction of CS proofs crushes the 4-round public-coin interactive argument system of Kilian [Kil92] into a single message by relying on the random oracle to serve both as a

CRHF (to compress a PCP proof as in [Kil92]), and in addition as a Fiat-Shamir-hash-functions to remove interaction as in [FS87]. Unfortunately, the possibility of implementing Fiat-Shamir-hash-functions by any hash function ensemble has been shown to be highly questionable [CGH04, DNRS03, GK03], and remains to be a significant open problem in cryptography. On the other hand, outside the random oracle model, no non-interactive delegation system is known for full polynomial time computations, even if the delegation system can use a specific, designated, verifier.

The possibility of non-interactive delegation is clearly most appealing. One may envision a delegator, using the computer facilities at a cloud to compute the result of an algorithm M, receives later the result via an e-mail with a very short "certificate" of correctness that can be written down in its entirty. Toward this goal, several works, starting with [GKR08] and followed by the work of Gennaro, Gentry and Parno [GGP10], and that by Chung, Kalai, and Vadhan [CKV10], have considered a relaxation in which the verifier is a designated verifier. Namely, the verifier may register as a client of the worker and establish some information in an off-line stage (or even engage in an interactive protocol with the worker in an off-line stage) which will enable him to verify the correctness of the proofs provided by the prover later in an on-line stage. Intuitively, what distinguishes a designated verifier from any other algorithm is that it knows some *secret* which will enable him (and only him) to verify the correctness of the proofs and catch a cheating prover (who does not know the secret) if it tries to prove a false statement.

Unfortunately, even with the relaxation of using a designated verifier, previous works do not match CS proofs. Let T be the worst case complexity of computing M. The construction of [GKR08] consists of one-time off-line stage with delegator complexity $\mathrm{poly}(\log T, k)$ in which the delegator posts one message to the worker of size $\mathrm{poly}(\log T, k)$; then for every input x the on-line stage is non-interactive (the worker sends a certificate σ of size $\mathrm{poly}(\log T, k)$ certifying the value of $M(x)$ which the delegator can verify), and is a function of the instance-based efficiency of the worker. But, it can only handle delegation of uniform $\mathcal{NC}$-computable algorithms. The designated verifier delegation schemes of [GGP10, CKV10] handle any P time computation. However, they consist of a one-time off-line stage with delegator complexity $\mathrm{poly}(T, k)$, and then an on-line two-round interactive protocol of comunication complexity $\mathrm{poly}(\log T, k, n)$. Thus, they only achieve *amortized* delegator computation complexity (and communication complexity) $\mathrm{poly}(\log T, k, n)$ over many inputs x. The computational complexity of the worker in [GGP10, CKV10] depends on the worst-case complexity of the algorithm being delegated rather than instance-based complexity.

Our first delegation system is a *non-interactive* designated verifier delegation scheme for all computation in $\mathcal{P}$. Additionally, the prover complexity is instance-based $\mathrm{poly}(k, t_x)$. The delegators complexity in the off-line and on-line stages are respectively is $\tilde{O}(n) \cdot \mathrm{poly}(\log t, k)$ and $\tilde{O}(n) \log \mathrm{poly}(\log T, k)$. Thus, the complexity guarantee is per input (not amortized).

The soundness of our protocol is based on the assumption that leveled Fully Homomorphic Encryption (FHE) schemes exist, the intractability of DDH, and a new Knowledge of Exponent Assumption (KEA), called q-KEA. The recent result of Brakersky and Vaikuntanathan [BV11], followed by Gentry [Gen11], provides a leveled FHE scheme (that is, the scheme can homorphically evaluate over ciphtertexts any circuits of depth bounded by a prior polynomial) based on the intractability of the Learning With Errors (LWE) assumption[2]. In fact, rather than assuming FHE, our construction can easily be transformed to use a PIR with receiver computation and communication complexities poly-logarithmic in the size of the database. For simplicity, we use FHE for the presentation.

[2]To achieve full homomorphism (without a prior bound on the depth of the circuits that the homomorphic evaluation procedure can take as input), Brakersky and Vaikuntanathan [BV11] need to additionally assume that their scheme is circular secure, that is, it is secure to encrypt the secret key using the encryption scheme itself. However, for our application, the function to be homomorphically evaluated is fixed and thus it suffices to use a leveled FHE scheme.

The knowledge of exponent assumption was originally introduced by Damgard [Dam91], and later extended by Bellare and Palacio [BP04] for showing the existence of 3-round zero-knowledge proofs. Our q-KEA assumption generalizes their assumptions so that we say q-KEA holds for a group G, if it is infeasible for a challenger, given a random generator $g = g_0$, q random elements $g_1, \ldots, g_q$ in the group, along with their α^{th} powers (i.e., $g_0^\alpha, \ldots, g_q^\alpha$), to generate a pair $c, \hat{c} = c^\alpha$, without actually "knowing" coefficients $a_0, \ldots, a_q$ such that $c = \prod_{i=0}^{q} g_i^{a_i}$. Here, "knowing" is captured by requiring that there exists a non-black-box extractor, who can depends on the code of the challenger, to extract out these a_i's. Our assumption is similar and in fact inspired by the assumption made by Groth [Gro10]. However, the actual assumption is incomparable with that of Groth.

Theorem 1 (Informal, Non-Interactive Designed Verifier Delegation Scheme) Assume the existence of a leveled fully homomorphic encryption scheme, the intractability of DDH and that q-KEA holds. Then, there exists a delegation scheme (D, W) with the following properties: Let k be a security parameter,

1. For a polynomial time computable function (algorithm) $M : \{0,1\}^n \to \{0,1\}^*$ of worst casr complexity T, the delegator in an off-line stage sends to the worker one message and keeps some secret information for later on-line stage. The computation complexity of the delegator in the off-line stage is $\tilde{O}(n)\text{poly}(k, \log T)$ and the commnication complexity is $\text{poly}(k, \log T)$

2. For any input $x \in \{0,1\}^n$, the worker in an on-line stage sends the output of the computation $y = M(x)$ and a proof of size $\text{poly}(k, \log t_x)$, where t_x is the running time of M on x. Furthermore, the worker runs in time $\text{poly}(k, t_x)$ and the delegator runs in time $\tilde{O}(n)\text{poly}(k, \log t_x)$.

Compared with the original CS proofs of Micali, proofs of the delegation scheme of Theorem 1 can only be verify by a verifier who holds the secret information generated in the off-line stage, whereas the CS proofs are publically verifiable in the random oracle model. Therefore, we also call such proof system *a designated verifier CS proof system.*

There is one issue with the soundness of our designated verifier delegation schemes which seems – so far– to be an inherent problem in all existing (including [GKR08, CKV10, GGP10]) designed verifier delegation schemes: the soundness of the system holds only under the restriction that the cheating worker does not learn whether the verifier accepts or rejects previous proofs from the prover. Alternatively stated, as soon as the delegator rejects one proof from the worker (and the worker is aware of that), soundness can no longer be claimed. The problem is that the worker can deviate from the protocol and generates messages to the verifier "improperly" so that by learning the verdicts of the verifier as a feedback, it can learn some information about the secret held by the designated verifier, which will then enable it to prove invalid statements later. We note that this is reminiscent of the problem encountered in the context of chosen ciphertext secure encryption schemes: Access to a decryption protocol with improperly formed ciphertexts may reveal information about the secret key and thus can compromise the security of the scheme. In the context of delegation, we call this the *verifier rejection* problem.

1.1.2 Solution 2: Two-Message Interactive Argument System

Our second solution is a two-message delegation scheme for all of $\mathcal{P}$ that addresses the verifier rejection problem. Compared with our first scheme (which does suffer from the verifier rejection problem), this scheme is interactive (consisting of two messages) and the delegator only has worst-case complexity (the worker still has instance-based complexity). More formally, we obtain:

Theorem 2 (Informal, Delegation Scheme without Verifier Rejection Problem) Assume the existence of a leveled fully homomorphic encryption scheme, the intractability of DDH, and that q-KEA holds. Then, there exists a delegation system (D, W) with the following properties: Let k be a security parameter,

1. For a polynomial time computable function (algorithm) $M : \{0,1\}^n \to \{0,1\}^*$ of worst case complexity T, and an input $x \in \{0,1\}^n$, the worker and the delegator interacts with each other in a 2-rounds protocol. The communication complexity of the protocol is $\text{poly}(k, \log T)$. The computational complexity of the worker and the delegator are respectively, $\text{poly}(k, t_x)$ and $\tilde{O}(n)\text{poly}(k, \log T)$, where t_x is the running time of M on x.

2. Soundness holds even against a cheating worker that is aware of the verdicts of the delegator. That is, it is infeasible for a cheating worker to convince the honest verifier to accept an incorrect output $y \neq M(x)$ even if the worker learns whether the delegator accept or reject previous proofs for other inputs.

Let us briefly compare our interactive solution to previous interactive arguments which can be used as delegation systems. Kilian [Kil92, Kil95] gives an *4-round* argument system for any $\mathcal{NP}$ computation, with communication complexity that is polylogarithmic, and verifier runtime which is linear in the input length (up to polylogarithmic factors), This is achieved by a constant round (four rounds) protocol, in which the prover first constructs a PCP for the correctness of the computation, and then Merkle-hashes it down to a short string and sends it to the verifier. To prove the soundness of his scheme, Kilian must assume the existence of strong collision-intractable hash functions: where collisions cannot be formed in sub-exponential time[3].

For uniform $\mathcal{NC}$ computations, [GKR08] shows how to combine their interactive proofs with a technique of Kalai and Raz [KR08] to give a two round delegation system for polynomial time bounded provers for $\mathcal{NC}$ computations based on the existence of a private information retrieval system (PIR) which achieves poly-logarithmic communication. Such PIR were first proposed by Cachin, Micali, and Stadler [CMS99] based on the ϕ-hiding assumption and most recently by Brakersky and Vaikuntanathan [BV11] based on the intractability of the LWE assumption.

We note that a 2008 paper by Di Crescenzo and Lipmaa [CL08] proposes a two-round argument with prover and verifier complexity and communication complexity as required for a delegation solution, but they do not prove soundness based on a computational assumption but rather, essentially their soundness proof assumes a restriction on the cheating prover: essentially if an efficient cheating prover can convince the verifier to accept the statement $x \in L$ it must be able to reconstruct a PCP proof of the statement. The question of two-round delegation argument systems for general $\mathcal{P}$ time computations and for (unrestricted class) polynomial time adversaries under computational assumptions thus remained open.

1.2 Our Techniques

The key tool that enables our construciton is an *extractable collision resistant hash function.* A collision resistent hash function (CRHF) is, informally speaking, a polynomial-time computable function $\mathcal{H}$ mapping binary strings of arbitrary length into reasonably short strings, so that it is computationally infeasible to find any collision (for $\mathcal{H}$), that is, any two different strings x and y for which $\mathcal{H}(x) = \mathcal{H}(y)$. In this work, we will consider CRHF that are additionally "extractable": Given a hash value, a preimage can be extracted efficiently. However, in general, such a construct cannot exist, since CRHF is one-way. To circumvent this problem, we will consider *non-black-box* extraction, that is, a preimage of a hash value can be extracted if the extractor "sees" the "code" of

[3]With standard intractability assumptions, one could get arguments of linear size communication complexity.

the machine that outputs that hash value. However, this relaxation alone is not sufficient to allow us to circumvent the impossibility. Consider, for instance, a hash function whose range is the set of all k-bit binary strings. In this case, a random hash value can be generated by simply tossing k random coins. But, given the code of such a sampling algorithm does not help invert the hash value at all. Instead, we require that values output by an extractable CRHF have some special structure; then, only for those *valid* hash values, a preimage can be extracted (in a non-black-box way).

Not surprisingly, we can construct such an extractable CRHF based on the q-KEA assumption. Our construction is similar to the construction of commitments with knowledge by Groth [Gro10], which in turn is based on a variant of the Pedersen commitment scheme. Let G be a group in which DDH and q-KEA holds, meaning that given properly sampled $g_0, \ldots, g_q$ in G, along with their α^{th} powers $\hat{g}_0, \ldots, \hat{g}_q$, if a challenger generates a pair $c, \hat{c} = c^\alpha$, then, there exists an extractor that extracts coefficients $a_0, \ldots, a_q$ such that $c = \prod_{i=0}^{q} g_i^{a_i}$. Then to hash a vector of elements $e_0, \ldots, e_q$, our CRHF simply outputs a pair $c = \prod_{i=0}^{q} g_i^{a_i}$ and $\hat{c} = \prod_{i=0}^{q} (\hat{g}_i)^{a_i}$. The collision resistence property of the hash function follows from the DDH assumption, and the extractability follows from q-KEA, since every valid output of the hash function satisfies that $\hat{c} = c^\alpha$, then by q-KEA, a pre-image $e_0, \ldots, e_q$ can be extracted. Given such an extractable CRHF, next we provide an overview of our construction of delegation schemes.

1.2.1 Overview of Our Approach

Similar to CS proofs in [Mic00], the overall approach to our construction is also to collapse rounds of the 4-message public-coin zero-knowledge argument of Kilian [Kil92], which in turn relies on PCP proofs and the notion of Merkle trees. So let us start by briefly reviewing them:

Probabilistic Checkable Proofs: Loosely speaking, a probabilistically checkable proof system (PCP) for a language consists of a probabilistic polynomial-time verifier having direct access to individual bits of a binary string. This string (called oracle) represents a proof, and typically will be accessed only partially by the verifier. Queries to the oracle are positions on the bit string and will be determined by the verifier's input and coin tosses. The verifier is supposed to decide whether a given input belongs to the language. If the input belongs to the language, the requirement is that the verifier will always accept given access to an adequate oracle. On the other hand, if the input does not belong to the language, then the verifier will reject with probability at least 1- ε for some small error bound ε, no matter which oracle is used. A formal definition of PCP can be found in section 3.3. We care about the complexity of the PCP verifier, in particular, the number of random coins it tosses $p_r(n)$ and the number of queries $p_q(n)$ it issues on input a statement of length n. Below we use polynomials p_r and p_q to denote the complexity of the PCP verifier.

Merkle Trees: A Merkle tree [Mer89] is a binary tree whose nodes are associated with values. A leaf node can store any value, but each internal node stores a value that is the hash of the concatenation of the values in its children through a collision-resistent hash function $\mathcal{H}$. Thus, if $\mathcal{H}$ produces k-bit outputs, each internal node of a Merkle tree, including the root, stores a k-bit value.

The crucial property of a Merkle tree is that, unless one succeeds in finding a collision for $\mathcal{H}$, it is computationally hard to change any value in the tree without also changing the root value. This property allows a party A to "commit" to n values, $v_1, \ldots, v_l$ (for simplicity assume $l = 2^d$ for some integer d), by means of a single k-bit value. That is, A stores value v_i in the i^{th} leaf of a full binary tree of depth $d = \log l$, and uses a collision-free hash function $\mathcal{H}$ to build a Merkle tree, thereby obtaining a k-bit value, rv, stored in the root.

More interestingly, A may "prove" what a particular v_i was "locally" by revealing just $d + 1$

values: v_i together with its *authentication path* ap_i, that is, the values stored in the siblings of the nodes along the path from leaf i (included) to the root (excluded), $Y_1, \ldots, Y_d$. It follows from the collision resistance property of $\mathcal{H}$ again that it is infeasible for A to come up with two different pairs of values and authentication paths that are both consistent with rv.

Kilian's Construction: In [Kil92] Kilian presents a special zero-knowledge argument for $\mathcal{NP}$, (P_k, V_k), exhibiting a polylogarithmic amount of communication, where prover P_k uses a Merkle tree in order to provide to V_k "virtual access" to a PCP proof. Below we describe the Kilian's protocol disregards the ZK property; our description is based on that presented in [BG08].

To prove a statement $x \in L$, the verifier V_k starts by sending the prover a CRHF $\mathcal{H}$ with output length k. The prover on private input a witness w, constructs a PCP-proof π. In order to yield efficient verifiability, P_k cannot send V_k the witness w nor π. Rather, P_k builds a Merkle tree with the proof π as the leaf values (using the collision-free hash function $\mathcal{H}$ from the verifier) producing a k-bit root value rv. It then "commits" itself to π by sending rv to the verier V_k. Since the Merkle tree is a full binary tree, the depth of the tree generated will be $d = \log |\pi|$; as the length of the proof is in turn bounded polynomially by the length of the witness, we have $d = O(\log |w|)$. The verier V_k then tosses $p_r(|x|)$ random coins r and sends them to the prover. Both the prover P_k and the verifier V_k computes the queries $q_1, \ldots, q_s$ $(s = p_q(|x|))$ by internally running the PCP verifier on input x and r. The prover P_k answers those queries by sending back answers $a_1, \ldots, a_s$ together with their authentication paths $ap_1, \ldots, ap_s$. V_k then checks whether a_j's authentication path is consistent with rv, and, if so, it is assured that a_j is the original value because the prover, being polynomial-time, cannot find a collision for $\mathcal{H}$. Finally, if any query answers and its corresponding authentication path are not consistent with rv, V_k rejects. Otherwise, it runs the PCP verifier on these answers $(x, a_1, \ldots, a_s)$ and accepts if and only if the PCP verifier accepts. Because V_{pcp} only makes $p_q(|x|) = \text{poly}(|x|)$ queries and each query will be answered by $kd = O(k \log |w|)$ bits of authentication path, the overall amount of communication is polylogarithmic in $|w|$.

At a very high-level, the soundness of Kilian's protocol follows from the fact that the Merkle tree provides the verifier "virtual access" to the PCP proof, in the sense that given the root value of the Merkle tree, for every query q, it is infeasible for a cheating prover to answer q differently depending on the queries. Therefore, interacting with the prover is "equivalent" to having access to a PCP proof oracle. Then it follows from the soundness of the PCP system that Kilian's protocol is sound.

Our Approach: We collapse rounds of the Kilian's protocol in two steps. First, we "compress" the two verifier's messages into one, which turns the protocol into a two-message protocol (P_2, V_2), using a leveled FHE scheme and assuming DDH and q-KEA. This two-message protocol essentially yields our second delegation solution. It does not suffer from the verifier rejection problem since it does not have an off-line stage and thus the delegator does not rely on any secret information. Therefore, even if the worker learns the verdicts of the verifier, soundness still holds for the soundness of interactive arguments is closed under sequential composition.

More precisely, let $E = (\text{KeyGen}, \text{Enc}, \text{Dec})$ be a leveled FHE scheme. To prove a statement x, the prover P_2 and the verifier V_2 proceed as follows:

Verifer's message: V_2 samples a random public and private key pair, $(pk, sk) \leftarrow \text{KeyGen}(1^k, 1^L)$ (where L is polylogarithmic in a bound on the length of the PCP proof of x), and a random $p_r(|x|)$-bit string r that will work as the random tape of the PCP verifier. It then sends a CRHF $\mathcal{H}$, pk and an encryption c of r under pk, i.e., $c = \text{Enc}_{pk}(r)$.

Prover's message: The prover P_2 first computes the PCP proof π and builds a Merkle tree $\mathcal{T}$ using $\mathcal{H}$ from π as in Kilian's protocol, producing a root value rv. It then tries to compute the PCP queries $(q_1, \ldots, q_s)$ based on the input x and the random coins r; however, since it

only receives an encryption c of r, it cannot evaluate $q_1, \ldots, q_s$ directly. Instead, it computes those queries *fully homomorphically* over x and c, producing encryptions c' of $q_1, \ldots, q_s$. It further computes answers $a_1, \ldots, a_s$ and the corresponding authentication paths $ap_1, \ldots, ap_s$ again fully homomorphically (over π, $\mathcal{T}$, and c'), yielding encryptions c''. Finally it sends back rv and c''.

Verifier's decision: The verifier after receiving rv and c'', decrypts c'' to obtain $a_1, \ldots, a_s$ and $ap_1, \ldots, ap_s$. It then runs the same decision procedure as V_k in Kilian's protocol.

Given the two-message protocol (P_2, V_2), notice that the verifier's message is almost "oblivious" of the input x, and only depends on its length $|x|$ (for deciding a bound on the length of the PCP proof for x and the number of coins tosses needed). Thus the first message can be generated beforehand by a trusted third party and the two-message protocol can be turned into a designated verifier delegatio system (P_1, V_1)—or a desigated verifier CS proof system—yielding our first delegation solution. More precisely, the delegator V_1 in the off-line stage generates a message $(\mathcal{H}, pk, \mathrm{Enc}_{pk}(r))$ as V_2 does in the first round; V_1 sends the message to the worker and keeps r and the secret key sk as its secret information. Later in the on-line stage, the worker P_1 proceeds the same as P_2 does in the second round; then the delegator decides whether to accept or reject as V_2 does.

The soundness of the designated verifier CS proof (D, P, V) follows directly from the soundness of the protocol (P_2, V_2). Intuitively, it seems that the latter should follow from the semantically security of the level FHE scheme. By the soundness of the PCP system, it is infeasible for a cheating prover to generate a proof for a false statement that convinces an honest PCP verifier (except from negligible probability). Furthermore, by the semantic security of the leveled FHE scheme, even if the cheating prover additionally receives an encryption of the random tape of the PCP verifier, it is still infeasible to generate an accepting proof. Then, as argued above for Kilian's protocol, since the Merkle tree provides "virtual access" to the proof, the cheating prover fails with overwhelming probability. The above argument is indeed very tempting. However, we do not manage to make it go through. The problem lies in the last part of the argument: unlike the case in Kilian's protocol where the verifier gets direct "virtual access" to the PCP proof, in our protocol, the verifier gets *encrypted* answers to the PCP queries and their corresponding authenticating paths back. Then suddenly, the cheating prover may be able to create some "rogue" connection between the encryption of the verifier's random tape, the root value of the Merkle tree and its encrypted answers.

To overcome this problem, we instantiate our designated verifier CS proof system with an extractable CRHF. We claim that such a proof system is sound. At a very high level, assume for contradiction that there exists a cheating prover that can prove a false statement, that is, given an encryption of the randomness of the PCP verifier, it manages to produce a root value rv, sends back encrypted answers to the PCP verifier's queries together with authentication paths (also encrypted) that are consistent with rv, and its answers convince the PCP verifier. Since the CRHF is extractable, one can extract out the values associated with the children of the root that hash to rv. Furthermore, by applying the extractability property recursively, we can further, level by level, extract out the values associated with the whole Merkle tree that hashes to rv. It then follows from the collision resistence property of the hash function that this extracted tree must "agree" with the prover's answers, that is, the authentication paths sent by the prover must be the same as the corresponding paths in the tree. Thus we obtain a machine that (given an encryption of the randomness of the verifier), outputs convincing answers to PCP queries *in plaintext*. This contradicts with either the soundness of PCP or the semantic security of the fully homomorphic encryption scheme.

There is one caveat in the argumentment above. When applying the extractablity of the CRHF to extract out a Merkle tree from a root value rv, the time of extraction soon explode. Consider,

for instance, that if extracting the preimage of a hash value output by a t-time machine takes time t^2. Then, after applying extraction recursively for d times, the total extraction time grows double exponentially to t^{2^d}. In comparison, we need to keep the extraction time "tamed"—in polynomial time—so that the FHE scheme remains semantically secure. We resolve this problem by utilizing the fact that our extractable CRHF has a very high "compression" rate: The ratio between the length of the input and that of the output is k (assuming q-KEA with $q = 2k-1$). Therefore, we can build a very "flat" Merkle tree by allowing each internal nodes to have k children. In this way, we can "commit" to any polynomial length string (or proof for delegation) into a root value of a constant depth Merkle tree. As a result, the extraction time for such a Merkle tree is still bounded by a polynomial. Finally, we remark that since the extraction time is bounded by a polynomial, it suffices to require our extractable CRHF to be collision resistant against polynomial time adversaries. On the other hand, Kilian's 4-round interactive argument system rely on a CRHF that is collision resistant against even sub-exponential adversaries. On a very high level, this difference stems from the fact that in our security proof, we use a strategy that finds a collision in polynomial time (by comparing the encrypted authentication paths from the cheating prover with that in the extracted Merkle tree from the root value from the prover), whereas the collision-finding strategy in Kilian's proof runs in sub-exponential time.

1.3 Other Related Works

Scaling down classical works on interactive proofs to address polynomial time languages result in non-polynomial time provers. In particular, the work of Lund, Fortnow, Karloff and Nissan [LFKN92] and that of Shamir [Sha92], the honest prover runs in super-polynomial time even for *log*-space languages. The work of Fortnow and Lund [FL93], using algebraic methods extending [LFKN92, Sha92], does explicitly address the question of interactive proofs for polynomial time languages and in particular $\mathcal{NC}$. Their protocol, however, has a non-polynomial time prover as in [LFKN92, Sha92].

The more recent work of Goldwasser, Kalai, and Rothblum [GKR08] provided an interactive proof to the delegation problem with a polynomial time prover, and verifier computation and communication complexity poly-logarithmic in the time of the computation, for uniform $\mathcal{NC}$ algorithms. Their protocol requires poly-logarithmic rounds of communication for $\mathcal{NC}$ computations. More generally, for general (non $\mathcal{NC}$) uniform circuit families, they provide an interactive protocol where the verifier time complexity, round complexity, and total communication complexity is a polynomial in the depth of the circut rather than its size.

The work of Babai, Fortnow, Levin and Szegedy [BFLS91] on Holographic Proofs for $\mathcal{NP}$—namely, PCP-proofs where the input is assumed to be presented to the verifier in an error-correcting-code format—raise similar complexity goals as in the delegation problem, requiring super-efficient verifiability (linear time in the input length and poly-logarithmic in the computation time), and efficient provability (polynomial time in the non-deterministic time complexity of accepting the input). However, the model of PCP in which they can obtain their results does not provide a solution to the delegation problem, as it proves soundness only against non-adaptive provers. Let us elaborate. [BFLS91] shows how to achieve verification time that is polylogarithmic in the length of the computation (on top of the time taken to convert the input into an error correcting code format), and a PCP-proof-string of length close to the computation time itself. However the soundness of proofs in the PCP model (as well as its more efficient descendants [PS94, BSGH$^+$06, BSGH$^+$05, Din07]) *requires* that the verifier/delegator will either 'poses' the entire PCP-proof-string (although the verifier will only read a few bits of it), or somehow have a guarantee that the prover/delegatee cannot change any bit of the PCP-proof-string after the verifier has started to request bits of it. Such guarantee is not achievable over a network as required in the delegation

		off-line		on-line			
Ref	Assumption	# msgs	D complexity	# msgs	D complexity	W complexity	Rejection
GKR08	none	0	0	$\text{poly}(d, \log t)$	$\text{poly}(n, d, \log t)$	$\text{poly}(t)$	No
BFL91, BFLS91	none	0	0	1	$\text{poly}(n, \log t)$	$\text{poly}(t)$	No
Kil92, Mic00	CRH	0	0	4	$\text{poly}(k, n, \log t)$	$\text{poly}(k, t)$	No
Kil92, Mic00	RO-Heur	1	$\text{poly}(k)$	1	$\text{poly}(k, n, \log t)$	$\text{poly}(k, t)$	No
GKR08, KR09	PIR	1	$\text{poly}(k)$	1	$\text{poly}(k, n, d, \log t)$	$\text{poly}(k, t)$	No
GGP09	FHE	1	$\text{poly}(k, T)$	2	$\text{poly}(k, n, \log T)$	$\text{poly}(k, T)$	Yes
CKV10-Thm5	FHE	0	$\text{poly}(k, T)$	2	$\text{poly}(k, n, \log T)$	$\text{poly}(k, T)$	Yes
CKV10-Thm6	FHE	4	$\text{poly}(k, n, \log T)$	2	$\text{poly}(k, n, \log T)$	$\text{poly}(k, T)$	Yes
Del_1	PIR, DDH, q-KEA	1	$\text{poly}(k, n, \log T)$	1	$\text{poly}(k, n, \log t)$	$\text{poly}(k, n, t)$	Yes
Del_2	PIR, DDH, q-KEA	0	0	2	$\text{poly}(k, n, \log T)$	$\text{poly}(k, n, t)$	No

Table 1: Results on Delegating Computation. D = delegator/verier, W = worker/prover, k = security parameter. Parameters of computation f being delegated: n = input length, T = maximum runnning time of f, $t = t_x$ = running time of f on input x, d = depth/parallel time (we assume $n \leq T \leq 2^d$).

setting.

Another related work is the recent work Chung, Kalai, Liu and Raz [CKLR11], relying on the existence of FHE scheme, addresses the question of memory delegation. Here, a delegator sends the entire content of its memory to a worker in an initial stage and from there on can issue some editing commands and can quickly verify the result of computations done by the worker on the memory. Our protocol can be used on top of the memory delegation scheme to enable the efficient verification of $\mathcal{P}$ time computations on the memory either using our first or second solution.

A related construct to our extractable collision resistant hash functions is the one-way extractable functions of Canetti and Dakdouk [CD08] which required that any party that manages to compute a value in the range of the function "knows" a corresponding preimage.

We summarize the known results in Table 1.3.

2 Outline

In Section 3 we provide some preliminaries and definitions of assumptions and primitives that we rely on. In Section 4, we introduce the notion of extractable collision resistant hash funcitons and provide a construction based on a new knowledge of exponent assumption called q-KEA in Section 4.1. Using extractable collision resistent hash functions, we construct a designated verifier CS proof system in Section 5. Finally, we obtain two schemes for delegating computation from a designated verifer CS proof system in Section 6.

3 Preliminaries

Let N denote the set of all positive integers. For any integer $n \in N$, let $[n]$ denote the set $\{1, 2, \ldots, n\}$, We denote by $\{0,1\}^n$ the set of binary strings of length n, and $|x|$ the length of a binary string. We denote by $\mathcal{PPT}$ probabilistic polynomial time Turing machines. We assume familiarity with interactive Turing machines, denoted ITM, interactive protocols, and computational indistinguishability; the formal definitions of interactive protocols and comutational indistinguishability are provided in Appendix A.

3.1 The DDH Assumption

let $\mathcal{G}$ take a security parameter k written in unary and output a description of a group $(p, G) \leftarrow \mathcal{G}(1^k)$ such that p is a k bit prime and G is a cyclic group of order p. The DDH assumption holds for $\mathcal{G}$ if the following two ensembles are computationally indistinguishable.

- $\left\{(p, G) \leftarrow \mathcal{G}(1^k),\ g \leftarrow_R G - \{0\},\ (a, x) \leftarrow_R \mathbb{Z}_p^2\ :\ (g, g^x, g^a, g^{ax})\right\}_{k \in N}$
- $\left\{(p, G) \leftarrow \mathcal{G}(1^k),\ g \leftarrow_R G - \{0\},\ (a, x, r) \leftarrow_R \mathbb{Z}_p^3\ :\ (g, g^x, g^a, g^r)\right\}_{k \in N}$

The DDH assumption implies that for every polynomial q, the following ensembles are indistinguishable.

$$\Big\{(p, G) \leftarrow \mathcal{G}(1^k),\ g \leftarrow_R G - \{0\},\ a \leftarrow_R \mathbb{Z}_p,\ (x_1, \ldots, x_{q(k)}) \leftarrow_R \mathbb{Z}_p^{q(k)} :$$
$$(g, g^{x_1}, \ldots, g^{x_{q(k)}}, g^a, g^{ax_1}, \ldots, g^{ax_{q(k)}})\Big\}_{k \in N}$$

$$\Big\{(p,G) \leftarrow \mathcal{G}(1^k),\ g \leftarrow_R G - \{0\}\,,\ a \leftarrow_R \mathbb{Z}_p,\ (x_1,\ldots,x_{q(k)}) \leftarrow_R \mathbb{Z}_p^{q(k)},\ (r_1,\ldots,r_{q(k)}) \leftarrow_R \mathbb{Z}_p^{q(k)} : (g, g^{x_1},\ldots,g^{x_{q(k)}}, g^a, g^{r_1},\ldots,g^{r_{q(k)}})\Big\}_{k\in N}$$

3.2 Fully Homomorphic Encryption Schemes

A public-key encryption scheme E = (KeyGen; Enc; Dec) is said to be fully homomorphic if it is associated with an additional polynomial-time algorithm Eval, that takes as input a public key pk, a ciphertext $\hat{x} = \mathrm{Enc}_{pk}(x)$ and a circuit C, and outputs, a new ciphertext $c = \mathrm{Eval}_{pk}(\hat{x}; C)$, such that $\mathrm{Dec}_{sk}(c) = C(x)$, where sk is the secret key corresponding to the public key pk. It is required that the size of $c = \mathrm{Eval}_{pk}(\mathrm{Enc}_{pk}(x); C)$ depends polynomially on the security parameter and the length of $C(x)$, but is otherwise independent of the size of the circuit C. We also require that Eval is deterministic, and the the scheme has perfect correctness (i.e. it always holds that $\mathrm{Dec}_{sk}(\mathrm{Enc}_{pk}(x)) = x$ and that $\mathrm{Dec}_{sk}(\mathrm{Eval}_{pk}(\mathrm{Enc}_{pk}(x); C)) = C(x)$). For security, we simply require that E is semantically secure.

In a recent breakthrough, Gentry [Gen09] proposed a fully homomorphic encryption scheme based on ideal lattices. Following this, Dijk, Gentry, Halevi and Vaikuntanathan[vDGHV10] proposed an alternative construction based on the extended GCD assumption. Very recently, Brakerski and Vaikuntanathan [BV11] proposed a new scheme based solely on the (standard) Learning With Error (LWE) assumption, which is in turn based on the worst-case hardness of short vector problems on *arbitrary* lattices. Immediately following that, Gentry [Gen11] further simplied their construction and greatly improved the efficiency of fully homomorhpic encryptions schemes. In all these schemes, the complexity of the algorithms (KeyGen; Enc; Dec) depends linearly on the depth d of the circuit C, where d is an upper bound on the depth of the circuit C that are allowed as inputs to Eval. However, under the additional assumption that these schemes are circular secure (i.e., remain secure even given an encryption of the secret key), the complexity of these algorithms are independent of C.

Our designated verifer CS proofs rely on the existence of a level FHE scheme. For simplicity of our presentation, we assume that the FHE scheme has perfect completeness. We note that the FHE schemes of both [Gen09] and [vDGHV10] indeed have perfect completeness. Furthermore, even if the FHE scheme does not have perfect completeness, it only causes the resulting designated verifier CS proofs to have an additional negligible completeness and soundness error.

Below, when referring to a FHE, we mean a leveled FHE with perfect completeness.

3.2.1 Private Information Retrieval

Rather than assuming FHE, our construction can easily be transformed to use a PIR with receiver computation and communication complexities poly-logarithmic in the size of the database. In [BV11], such a PIR scheme is constructed from LWE-based FHE. Therefore PIR is, in some sense, a weaker assumption than FHE, and yet it suffices for our construction.

3.3 Probabilistically Checkable Proofs

Loosely speaking, a probabilistically checkable proof system (PCP) for a language consists of a probabilistic polynomial-time verifier having direct access to individual bits of a binary string. This string (called oracle) represents a proof, and typically will be accessed only partially by the verifier. Queries to the oracle are positions on the bit string and will be determined by the verifier's input and coin tosses. The verifier is supposed to decide whether a given input belongs to the language. If the input belongs to the language, the requirement is that the verifier will always

accept given access to an adequate oracle. On the other hand, if the input does not belong to the language, then the verifier will reject with probability at least 1- ε for some small error bound ε, no matter which oracle is used. Below we present the formal definition of PCP.

Definition 1 (Probabilistically Checkable Proofs)**.** *A probabilistically checkable proof system with error bound ε for a language L consists of probabilistic polynomial-time oracle machine (called verifier) V and two polynomials p_r, p_q satisfying*

Completeness: *For every $x \in L$ there exists an oracle π_x such that,* $\Pr\left[V^{\pi_x}(x) = 1\right] = 1$

Soundness: *For every $x \notin L$ and every oracle π,* $\Pr\left[V^{\pi}(x) = 1\right] \leq \varepsilon(|x|)$

Complexity: *On any input of length n, V makes at most $p_r(n)$ coin tosses and at most $p_q(n)$ oracle queries.*

In this work, we do not care about the refined complexity p_r and p_q of the PCP system. Instead, we will care about that a set $\mathcal{N} \subseteq \mathcal{NEXP}$ (the non-deterministic CS language defined in Section 5.1) has a PCP system with a negligible error bound, and satisfies the following additional proprieties defined in [BG08].

Definition 2 (Additional Properties for PCP [BG08])**.** *Let V be a PCP verifier with error bound ε and complexity p_r p_q, for a language $\mathcal{N} \subseteq \mathcal{NEXP}$, and let $\mathcal{R}$ be a corresponding witness relation. Consider the following properties.*

Relatively-efficient oracle-construction: *There exists a deterministic polynomial time machine P, such that, for every $(x, w) \in \mathcal{R}$,* $\Pr\left[\pi_x = P(x, w) \ : \ V^{\pi_x}(x) = 1\right] = 1$*. As a result, there exists a polynomial p_p such that the proof π_x has length bounded by $p_p(|x|, |w|)$.*

Non-adaptive Verifier: *The verifier algorithm V can be decomposed into a pair of algorithms Q and D, such that, on input x and a random tape r of length $p_r(|x|)$, the verifier runs $Q(x, r)$ to generate $s = p_q(|x|)$ queries $q_1, \ldots, q_s$, obtains answers $a_1, \ldots, a_s$ to those queries from the oracle, and decides by computing and outputting $D(x, r, a_1, \ldots, a_s)$.*

The above properties are known to hold for many PCP systems. Below, we consider only PCP system with the above properties, and denote it as $(P, V = (Q, D))$ with complexity p_p, p_r, p_q.

4 Extractable Collision Resistence Hash Function

In this section, we formally define the notion of extractable collision resistant hash functions (CRHF) and construct such a hash function based on a new knowledge of exponent assumption called q-KEA introduced in Section 4.1.

Definition 3 (Extractable Collision Resistence Hash Function)**.** *Let $\{I_k\}_{k \in N}$ be a sequences of sets of indexes. A tuple $\mathcal{H} = (\{H_{k,id}\}_{k \in N, id \in I_k}, \{\mathsf{valid}_k\}_{k \in N})$, is an extractable collision resistence hash function if $\{H_{k,id}\}$ is a family of collision resistent hash functions, $\{\mathsf{valid}_k\}$ with $\mathsf{valid}_k : I_k \times \{0,1\}^* \to \{0,1\}$ is a sequence of functions that decides the validity of a hash value, and satisfies the following:*

Extractability: *For every non-uniform deterministic polynomial time machine A, there exists a non-uniform deterministic polynomial time machine E and a negligible function μ, such that, for every $k \in N$ and every auxiliary input $z \in \{0,1\}^*$, the following holds.*

$$\Pr\Big[id \leftarrow I_k,\ \vec{v} = A(1^k, id, z);\ \vec{o} = E(1^k, id, z)\ :\ \exists j \in [|\vec{v}|],\ \mathsf{valid}_k(id, \vec{v}_j) = 1 \text{ and } H_{k,id}(\vec{o}_j) \neq v\Big] \leq \mu(k)$$

In other words, an CRHF is extractable if for every machine A there exists an extractor E that can extract out a pre-image for every valid hash value that A outputs. Next we proceed to construct an extractable CRHF. Our construction relies on a new knowledge of exponent assumption introduced below.

4.1 q-Knowledge of Exponent Assumption

The knowledge of exponent (KEA) assumption introduced by Damgård [Dam91] says that given g, g^α it is infeasible to create $c, \hat{c}$ so that $\hat{c} = c^\alpha$ without knowing a so $c = g^a$ and $\hat{c} = (g^\alpha)^a$. Bellare and Palacio [BP04] extended this to the KEA3 assumption, which says that given $g, g^x, g^\alpha, g^{x\alpha}$, it is infeasible to create c and $\hat{c}$ so that $\hat{c} = c^\alpha$ without knowing a, b such that $c = g^a(g^x)^b$ and $\hat{c} = (g^\alpha)^a, (g^{x\alpha})^b$. This assumption has been used also in bilinear groups by Abe and Fehr [AF07] who called it the extended knowledge of exponent assumption.

Our q-Knowledge of Exponent (q-KEA) Assumption generalizes the KEA and KEA3 assumptions in the following aspect: Instead of receiving only a pair of elements g, g^x and their α^{th} powers, a challenger now is given q random elements $g, g^{x_1}, \ldots g^{x_q}$ and their corresponding α^{th} powers. The assumption states that it is infeasible to create c and $\hat{c}$ so that $\hat{c} = c^\alpha$ without knowing $a_0, a_1, \ldots, a_1$ such that $c = g^{a_0} \prod_{i=1}^{q} g^{x_i a_i}$. A similar assumption, called the q-power knowledge of exponent assumption for bilinear groups, was introduced by Groth [Gro10] for constructing a short (sublinear length) pairing-based Non-interactive Zero-Knowledge Arguments. Their assumption is almost the same except that the challenger receives q elements of a special form $g, g^{x_1}, \ldots g^{x_q}$ and their α^{th} powers. We note that the two assumptions are incomparable.

Definition 4 (q-Knowledge of Exponent Assumption)**.** *Let q be any polynomial. The q-KEA assumption holds for a family of groups $\mathcal{G}$ if for every deterministic polynomial time machine A, there exists a deterministic polynomial time machine E and a negligible function μ, such that, for every $k \in N$ and every auxiliary input $z \in \{0,1\}^*$, the advantage of A in the following experiment $Exp^q_{A,E}(k, z)$ is bounded by $\mu(k)$.*

Experiment $Exp^q_{A,E}(1^k, z)$**:**

1. *Let $(p, G) = \mathcal{G}(1^k)$, g_0 a random generator in G, $g_1 \ldots g_{q(k)}$ random elements in G, and α a random element in $\mathbb{Z}_p$; let $\delta = (g_0, \ldots, g_{q(k)}, g_0^\alpha, g_1^\alpha, \ldots, g_{q(k)}^\alpha)$.*
2. *A, on inputs $1^k, \delta$ and auxiliary input z, outputs $(s, (c_1, \hat{c}_1), \ldots, (c_s, \hat{c}_s))$*
   E, on inputs $1^k, \delta$ and auxiliary input z, outputs s' and a sequence of exponents $(a_0^j, \ldots a_{q(k)}^j)$ for $j \in [s']$.
3. *The experiement outputs 1 if and only if there exists a $j \in [s]$, such that, $\hat{c}_j = c_j^\alpha$ but $c_j \neq \prod_{i=0}^{q(k)} g_i^{a_i^j}$.*

The advantage of A in the above experiment equals to the probability that the experiment outputs 1.

4.2 Constructing Extractable CRHF

Our construction of extractable CRHF is essentially the same as the construction of commitments with knowledge by Groth [Gro10], which in turn is based on a variant of the Pedersen commitment scheme. Consider the following CRHF:

$\mathcal{H} = \left(\{H_{k,id}\}_{k \in N, id \in I_k}, \{\mathsf{valid}_k\}_{k \in N}\right)$:

Sampling: Let $(p, G) = \mathcal{G}(1^n)$ and $q = q(k)$. The k^{th} index set is defined as follows:

$$I_k = \Big\{ (g_0, \ldots, g_q, \hat{g}_0, \ldots, \hat{g}_n) \ : \\ g_0 \in G - \{0\}\,,\ g_1, \ldots, g_q \in G,\ \exists \alpha \in Z_p \text{ s.t. } \forall i,\ \hat{g}_i = g_i^{\alpha} \Big\}$$

In other words, to sample a random hash function $H_{k,id}$, simply sample $g_0, \ldots, g_k$ at random from G as specified above, sample α at ranodm from $\mathbb{Z}_p$, and output $id = (g_0, \ldots, g_q, g_0^{\alpha}, \ldots, g_q^{\alpha})$.

Hashing: For every k and $id = (g_0, \ldots, g_q, \hat{g}_0, \ldots, \hat{g}_q) \in I_k$, hash function $H_{k,id} : \mathbb{Z}_p^{q+1} \to G^2$, on input $(a_0, \ldots, a_q) \in Z_p^n$, outputs $c = \prod_{i=0}^{q} g_i^{a_i}$ and $\hat{c} = \prod_{i=0}^{q} \hat{g}_i^{a_i}$.

Verifying: On input a hash value $(c, \hat{c}) \in G^2$ and an index $id = (g_0, \ldots, g_q, \hat{g}_0, \ldots, \hat{g}_q) \in I_k$, the function valid_k outputs 1 if there exists a $\alpha \in \mathbb{Z}_p$ such that $\hat{g}_i = g_i^{\alpha}$ for all i and $\hat{c} = c^{\alpha}$.

The hash function $H_{k,id}$ maps $q+1$ elements in Z_p to two elements in G, achieving a "compression rate" of $(q+1)/2$. It follows from stardard technique that $\mathcal{H}$ is collision resistent assuming that the DDH assumption holds on $\mathcal{G}$; here we omit the proof.

Lemma 1. *Assume that the DDH assumption holds on $\mathcal{G}$. Then, $\{H_{k,id}\}_{k \in N, id \in I_n}$ is a family of collision resistent hash function.*

Furthermore, it follows directly from q-KEA that the hash function is also extractable. That is,

Proposition 1. *Let q be any polynomial. Assume that the DDH assumption and q-KEA hold on $\mathcal{G}$. Then, $\mathcal{H}$ is a extractable CRHF with compression rate $(q(k)+1)/2$.*

Proof. By construction, hash function $\mathcal{H}_{k,id}$ has compression rate $(q(k)+1)/2$. Then, following Lemma 1, it only remains to show that $\mathcal{H}$ is extractable. Fix any non-uniform deterministic polynomial-time machine A. It follows from q-KEA that there exists an extractor E (also non-uniform deterministic polynomial-time) such that, for every k, z, it holds that, except from negligible probability, whenever A on input a random $id = (g_0, \ldots, g_q, g_0^{\alpha}, \ldots, g_q^{\alpha}) \in I_k$, outputs a valid hash value $(c_j, \hat{c}_j = c_j^{\alpha})$, E on the same input, outputs $(a_0^j \ldots a_q^j)$ such that $c_j = \prod_{i=0}^{q} g_i^{a_i^j}$. Since $(a_0^j \ldots a_q^j)$ is a valid preimage for $(c_j, \hat{c}_j)$, E is a valid extractor for A w.r.t. $\mathcal{H}$. □

4.3 Extractable Merkle Trees

Recall that Merkle tree is a full binary tree whose nodes are associated with value generated in a special way according to a CRHF such that the value of an internal node is the hash of the concatenation of the values of its children. We show that by instantiating the Merkle tree with an extractable CRHF, we obtain a Merkle tree that is also *extractable*, that is, given only the root value rv of a Merkle tree, one can extract out a Merkle tree that is consistent with rv.

Recall that our extractable CRHF $\mathcal{H}$ achieves a high "compression rate" $(q(k)+1)/2$. When using such a CRHF, we can afford to let each internal node have more than two children. More precisely, a n-ary (for now, consider n to be even) Merkle tree is a n-ary tree built in the same way as a classical Merkle tree, except that each internal node can have up to n children and its associated value is the hash of the concatenation of the values of its n children through a hash function with "compression rate" n. As classical (binary) Merkle trees, the root value of a k-ary Merkle tree is a good "commitment" of the original values associated with the leaves, and a

particular value v_i can be "decommited" by revealing $d(n-1)+1$ values: v_i together with its authentication path ap_i consisting of the values stored in the $d(n-1)$ siblings of the nodes along the path from leaf i (inclusive) to the root (exclusive). To check the consistency of a pair of value v_i and authentication path ap_i with a root value rv, the verify procedure proceeds as follows: On input v_i and $ap_i = (Y_1, \ldots, Y_d)$ where each Y_j consists of $n-1$ values $Y_j^1, \ldots, Y_j^{n-1}$, verify sets $X_1 = v_i$ and computes the values X_j for j from 2 to d by setting $X_{j+1} = \mathcal{H}(Y_j^1 \| \ldots \| Y_j^{t-1} \| X_j \| Y_j^t \| Y_j^{n-1})$ if i_j (the j^{th} bit of i in k-ary representation) is t; finally, verify checks whether $X_d = rv$ and outputs 1 if and only if this holds.

We claim that Merkle trees constructed using an extractable collision resisent hash function $\mathcal{H}$ is also "extractable". This means, for every efficient challenger A, and every constant D, there exists an efficient extractor E_D such that whenever A on input a random index id, outputs a valid root value rv, E_D on receiving the same index id outputs a labeled tree $\mathcal{T}$ of depth D that is consistent with rv. A tree $\mathcal{T}$ is consistent with a root value rv, if the following procedure outputs 1.

$\mathsf{merkle}_k(id, D, rv, \mathcal{T})$ check:

1. $\mathcal{T}$ has depth at most D.
2. The value associated with the root of $\mathcal{T}$ equals to rv.
3. The value v associated with every internal node equals to the hash of the concatenation of the values associated with its children.
4. The value v associated with every leaf on depth $d < D$ is *not* a valid hash value, that is, $\mathsf{valid}_k(1^k, id, v) = 0$. (A node is at depth d if its distance from the root is d.)

Output 1 if all the above conditions hold, and 0 otherwise.

Lemma 2. *Let $\mathcal{H} = \left(\{H_{k,id}\}_{k\in N, id\in I_k}, \{\mathsf{valid}_k\}_{k\in N}\right)$ be an extractable CRHF with compression rate l. Then for every non-uniform deterministic polynomial time machine A and every constant D, there exists a non-uniform deterministic polynomial time machine E_D and a negligible function μ_D, such that, for every $k \in N$ and every auxiliary input $z \in \{0,1\}^*$, the following holds.*

$$\Pr\Big[id \leftarrow I_k,\ rv = A(1^k, id, z);\ \mathcal{T} = E(1^k, id, z)\ :\ \mathsf{valid}_k(id, rv) = 1 \text{ and } \mathsf{merkle}_k(id, D, \mathcal{T}, rv) \neq 1\Big] \leq \mu(k)$$

Proof. This lemma follows essentially by applying the extractability property of the CRHF $\mathcal{H}$ recursively. Since the lemma only cares about extracting constant depth Merkle trees, we only need to apply the extractability property for a constant number of times and thus the resulting extractor for the Merkle tree is efficient. More precisely, fix any non-uniform deterministic polynomial time machine A; we prove by induction that for every constant D there exists an extractor E_D and a negligible function μ_D satisfying the lemma. When $D = 1$, it follows immediately from the extractability of the CRHF $\mathcal{H}$ that such E_1 and μ_1 exist. Then assuming that E_d and μ_d exist for $D = d$ a constant, we want to exhibit an extractor E_{d+1} and a negligible function μ_{d+1} for $D = d+1$. Fix any $k \in N$ and an auxiliary input $z \in \{0,1\}^*$. By our hypothesis, it holds that except from a μ_d fraction of indexes $id \in I_k$, it holds that whenever the output of rv of the challenger $A(1^k, id, z)$ is valid, the extractor $E_d(1^k, id, z)$ outputs a labeled tree $\mathcal{T}_d$ of depth bounded by d that is consistent with rv, (i.e., $\mathsf{merkle}(id, d, rv, \mathcal{T}_d) = 1$). Then consider a machine E'_d which simply runs E_d internally and outputs only the values associated with the leaves at depth d in $\mathcal{T}_d$ (output by E_d). By the extractability of $\mathcal{H}$, there exists a deterministic polynomial time machine X_d and

a negligible function μ such that, except from a μ fraction of $id \in I_k$, for every valid hash value v output by $E'_d(1^k, id, z)$, $X_d(1^k, id, z)$ outputs a corresponding preimage $(v_1 \dots v_{l(k)})$. Using X_d we construct E_{d+1} as follows. Machine $E_{d+1}(1^k, id, z)$ internally runs $E_d(1^k, id, z)$ and $X_d(1^k, id, z)$, obtaining outputs $\mathcal{T}_d$ and w resepctively; it then constructs a depth $d+1$ tree T_{d+1} by extending $\mathcal{T}_d$: For every leaf node of $\mathcal{T}_d$ at depth d, if w contains a valid preimage $(v_1 \dots v_{l(k)})$ of the value associated with it, extend that leaf with $l(k)$ children with values v_1 to $v_{l(k)}$ respectively. If $\mathcal{T}_d$ is consistent with rv and X_d extracts a pre-image for every valid hash value associated with nodes at depth d of $\mathcal{T}_d$, $\mathcal{T}_{d+1}$ is also consistent with rv. Therefore E_{d+1} "fails" only if E_d or X_d "fails". By our hypothesis and and the extractability of $\mathcal{H}$, we obtain E_{d+1} fails with probability at most $\mu_{d+1}(k) = \mu_d(k) + \mu(k)$. □

5 Designated Verifier CS Proofs

Our designated verifier CS proofs, like CS proofs, has efficient instance-based provability and verifiablity (resepctively, polynomial and polylogarithmically in the time of the computation to be proved). However, it is weaker in the following two aspects: First, proofs of our protocol are only verifable by a designated verifier (whereas CS proofs are publicly verifiable), and second, our protocol only achieves a weaker (computational) soundness guarantee, that is, soundness only holds if the (computationally bounded) cheating prover tries to prove an invalid *polynomial-time* computation (whereas no invalid, even exponential-time, computation can be proved using CS proofs).

5.1 Defining Designated Verifier CS Proofs

We first recall the definition of CS language introduced by Micali [Mic00]

Definition 5 (CS language [Mic00])**.** *The CS language, denoted by $\mathcal{L}$, is the set of all quadruples $q = (M, x, y, t)$, such that M is (the description of) a Turing machine, x and y are a binary strings, and t a binary integer such that it holds that $|x| \leq t$, $|y| \leq t$, $M(x) = y$ and $\mathsf{steps}(M(x)) = t$, where $\mathsf{steps}(M(x))$ denote the number of steps that M takes on input x.*

The CS language essentially includes all quadruples that correspond to a valid *deterministic* computation. In fact, the CS proofs are capable of proving all valid *non-deterministic* computation as well. Formally, it corresponds to the following language.

Definition 6 (Non-deterministic CS language)**.** *The non-deterministic CS language, denoted by $\mathcal{N}$, is the set of all quadruples $q = (M, x, t)$, such that M is (the description of) a Turing machine, x is a binary string, and t a binary integer, it holds that there exists a witness w, $|x| \leq t$, $|w| \leq t$, $M(x, w) = 1$ and $\mathsf{steps}(M(x, w)) = t$. Denote by $\mathcal{R}$ the witness relation of $\mathcal{N}$, that is,*

$$\mathcal{R} = \{((M, x, t), w) \ : \ |x| \leq t, \ |w| \leq t, \ M(x, w) = 1, \ \mathsf{steps}(M(x, w)) = t\}.$$

We also denote by $\mathcal{R}(X)$ the set of witnesses w such that $\mathcal{R}(X, w) = 1$.

In the designated verifier model, a non-interactive proof system has an associated polynomial-time sampleable distribution D over binary strings of the form (pp, sp). During a setup phase, a trusted party samples from D, publishes pp and privately hands the Verifier sp. The Prover and Verifier then use their respective values during the proof phase.

Definition 7 (Designated Verifier CS Proofs)**.** *A triple of algorithms, (D, P, V), is called a designated verifier CS proof system, if D is probablistic polynomial time, P and V are deterministic, the second of which runs in polynomial time, and the following properties hold:*

Feasible Completeness: *There exists polynomials t_P and ℓ, such that for every $X = (M, x, t) \in \mathcal{N}$ and every $w \in \mathcal{R}(x)$, every $k \in N$, and every possible output (pp, sp) of $D(1^k, 1^{|X|})$, $P(1^k, pp, X, w)$ halts within $t_p(k, |X|, t)$ computational steps, outputting a proof π of length smaller than $\ell(k, |X|)$. Furthermore, it holds that,*

$$\Pr\left[(pp, sp) \leftarrow D(1^k, 1^{|X|});\ \pi \leftarrow P(1^k, pp, X, w)\ :\ V(1^k, pp, sp, X, \pi) = 1\right]$$

Computational Soundness: *For every deterministic polynomial time machine P^*, there exists a negligible function μ, such that for every $k \in N$ and every n that is polynomially bounded by k, it holds that:*

$$\Pr\left[(pp, sp) \leftarrow D(1^k, 1^n);\ (X, \pi) \leftarrow P^*(1^k, pp)\ :\ X \notin \mathcal{N} \text{ and } V(1^k, pp, sp, X, \pi) = 1\right] \leq \mu(k)$$

The above defintion adapts the original notion of CS proofs [Mic00] into the designated verifier model. In this model, the property of public verifiability no longer holds; instead, only verifiers holding certain secret information sp can verifiy the validity of the proof. Next, we further relax the soundness property to only require it to hold when the cheating prover is trying to prove an invalid but *polynomial-time* computation.

Definition 8 (Weak Soundness)**.** *We say that a designated verifier CS proof system (D, P, V) has weak soundness if the computational soundess property is replaced by the following:*

Weak Computational Soundness: *For every deterministic polynomial time machine P^*, and every constant $c \in N$, there exists a negligible function μ, such that for every $k \in N$ and every n that is polynomially bounded by k, it holds that:*

$$\Pr\Big[(pp, sp) \leftarrow D(1^k, 1^n);\ (X, \pi) \leftarrow P^*(1^k, pp)\ :\ X = (M, x, t) \text{ and } t \leq k^c \text{ and } X \notin \mathcal{N} \text{ and } V(1^k, pp, sp, X, \pi) = 1\Big] \leq \mu(k)$$

Below, when referring to a designated verifier CS proof system, we mean such a proof system with weak soundness.

Remark 1. *We remark that in our definition of computational soundness, the cheating prover only sees the public string pp. This is different from the case in the definition of designated verifier non-interactive zero-knowledge proof system in the context of constructing CCA2 secure encryption schemes, where the soundness needs to hold even if the cheating prover has access to a verification oracle, which on input a statement and a proof tells the cheating prover whether this is an accepting proof or not. One consequence of this difference is that soundness of our proof system may not hold when a cheating prover participates in a game where it interacts with the honest verifier in many rounds proving many statements and may learn the decision of the verifier in each round. As we shall see later, this is relevant to the soundness of the delegation schemes that we construct from designated verifier CS proof systems.*

5.2 Constructing Designated Verifier CS Proofs

Our designated verifier CS proof system (D, P, V) for the non-deterministic CS language $\mathcal{N}$ rely on the following building blocks:

1. An extractable collision resistent hash functions $\mathcal{H} = (\{H_{n,id}\}_{n\in N, id\in I_n}, \{\mathsf{valid}_k\}_{k\in N})$ be with output length k and compression rate $l(k) = k$.
2. A fully homomorphic encryption scheme $E = (\text{KeyGen}, \text{Enc}, \text{Dec})$ with perfect completeness.
3. A PCP system $(P_{pcp}, V_{pcp} = (Q, D))$ with complexity p_p, p_r, p_q for $\mathcal{N}$, such that, the number of random coins tossed by the verifier on an input $X = (M, x, t) \in \mathcal{N}$ is $p_r(|X|) = \text{poly}(\log|X|, \log t)$, and the number of queries generated is $p_q(|X|) = \text{poly}(\log|X|, \log t)$.

Let k be a security parameter. To prove a statement $X \in \mathcal{N}$, our scheme proceeds as follows:

Set-up: D on input 1^k and $1^{|X|}$, do:

- Sample at random a hash function $H_{k,id}$, a $p_r(|X|)$-bit string r, and a pair of public and private keys of E, $(pk, sk) \leftarrow \text{KeyGen}(1^k, 1^L)$, where $L = \log(\text{poly}(p_p(|X|)))$ is a bound on the depth of the $\mathsf{section\ circuit}$ (introduced shortly below) on input vectors and trees of length $\text{poly}(p_p(|X|))$.
- Compute the encryption of r under pk, $c = \text{Enc}_{pk}(r)$;
- Output $pp = (id, pk, c)$ and $sp = (sk, r)$.

Prover's Message: P on input 1^k, $X = (M, x, t)$, a witness w, and $pp = (id, pk, c)$, do:

- Internally run P_{pcp} on input X and w to generate a PCP-proof π; the length of the proof is $|\pi| = p_p(|X|, |w|)$ bounded by $p_p(|X|, t)$ as $|w| \leq t$.
- Build a $l(k) = k$-ary Merkle tree $\mathcal{T}$ by setting the value associated with the i^{th} leaf node to be equal to π_i[4], and computing the values of the internal nodes using the hash function $H_{k,id}$; let rv be the root value generated and d be the depth of the Merkle tree;
- Evaluate homomorphically the circuit Q over X and c to compute the PCP queries, yielding $c' = \text{Eval}_{pk}(Q; (X, c))$;
- Select homomorphically the answers and their corresponding authentication paths according to c', yielding $c'' = \text{Eval}_{pk}(S; (\pi, \mathcal{T}, c'))$, where S is a $\mathsf{selection\ circuit}$ which on input a string str, a tree T, and a vector of indexes $\vec{q}$, returns for every $i \in [|\vec{q}|]$, $str_{q[i]}$ and the path leading from the root to the $q[i]^{\text{th}}$ leaf in T;
- Outputs $\pi = (rv, d, c'')$.

Verifier's Decision: V on input 1^k, $X = (M, x, t)$, $pp = (id, pk, c)$, $sp = (sk, r)$ and $\pi = (rv, d, c'')$, do:

- Verify that the depth of the Merkle tree is correctly bounded, that is, $d \leq \log_k(p_p(|X|, t))$.
- Let $s = p_q(|X|)$; Decrypt c'' using the secret key sk, yielding $(a_1, \ldots, a_s, ap_1, \ldots, ap_s) = \text{Dec}_{sk}(c'')$.
- Check for every $i \in [s]$, whether a_i, ap_i are consistent with rv; if for any $i \in [S]$, $\mathsf{verify}(id, a_i, ap_i) \neq 1$, reject and abort;
- Run D on input X, r, $a_1, \ldots, a_s$; accept if and only if D outputs 1.

[4]We assume w.l.o.g. that the length of the PCP proof π is a power of k. Otherwise, we can always make it to be the case by padding π with 0's.

We analyze the efficinecy of the above scheme as follows: By construction and the relatively efficient oracle construction property of the PCP system, the prover P runs in time $\text{poly}(k, |X|, t)$. It is easy to check that the set-up algorithm runs in time $\text{poly}(k, |X|)$ and the verifier runs in $\text{poly}(k, |X|, \log t)$. Finally, we bound the length of the public information pp and the proof generated by the prover. Since the PCP verifier tosses $p_r(|X|) = \text{poly}(\log |X|, \log t))$ number of coins, the length of the public information pp is bounded by $\text{poly}(k, \log |X|, \log t)$. Then, since the PCP proof has length $\text{poly}(k, |X|, t)$, the Merkle tree generated by the prover has depth $d = O(\log_k(k, |X|, t))$, and thus the length of the answers to the s PCP queries where $s = p_q(|X|) = \text{poly}(\log |X|, \log t)$ is $s(d(k-1)+1)k = \text{poly}(k, \log |X|, \log t)$. Therefore, the size of the proof is indeed $\text{poly}(k, |X|)$ as required in Definition 7.

Remark 2. *As we shall see later, when constructing delegation schemes using the above designated verifier CS proof system. We utilize the fact that the lengths of the public information pp and the CS proof generated are both in fact* $\text{poly}(k, \log |X|, \log t)$*, depending polylogarithmically in the length of the input. Correspondingly, this will yield a delegation scheme with communication complexity bounded poly-logarithmically in the complexity of the computation being delegated and length of the input x' of the computation, provided that the delegator and the worker receives the input x' from outside. From another perspective, this separates the communication complexity needed for transferring the input (from the delegator to the worker), from that needed for verifying the correctness of the output, which is only polylogarithmic in the input length.*

5.2.1 (D, P, V) works

Theorem 1. *(D, P, V) is a designated verifier CS proof system.*

Proof. The completeness of the system follows from the perfect completeness of the fully homomorphic encryption scheme and the completeness of the PCP system. The computation and communication complexity of the system satisfies the definition of designated verifier CS proofs as analyzed above. Thus it only remains to show that (D, P, V) satisfies weak soundness.

We need to show that no prover can prove a false statement that has a polynomial time bound. That is, for every constant C and every cheating prover P^*, there exists a negligible function μ, such that the following holds for every $k \in N$ and $n \in \{0,1\}^*$.

$$\Pr\Big[(pp, sp) \leftarrow D(1^k, 1^n);\ (X, \pi) \leftarrow P^*(1^k, pp)\ :\ X = (M, x, t) \text{ and } t \le k^c \text{ and } X \notin \mathcal{N} \text{ and } V(1^k, pp, sp, X, \pi) = 1\Big] \le \text{neg}(k) \quad (1)$$

Recall that a proof π of (D, P, V) consists of three components (rv, d, c''), where rv is the root value of a Merkle tree and d is the depth of the tree. Towards proving the weak soundness, we show that no prover can "cheat" by giving a proof with a constant depth d. That is, for every constant D and every cheating prover P^*, there exists a negligible function μ, such that the following holds

$$\Pr\Big[(pp, sp) \leftarrow D(1^k, 1^n);\ (X, \pi) \leftarrow P^*(1^k, pp)\ :\ \pi = (rv, d, c'') \text{ and } d \le D \text{ and } X \notin \mathcal{N} \text{ and } V(1^k, pp, sp, X, \pi) = 1\Big] \le \mu(k) \quad (2)$$

We claim that if the above holds, then so does weak soundness. Assume for contradiction that the above holds but there exits a cheating prover P^* and a constant C such that (1) is false. Recall that on input a statement $X = (M, x, t)$ and $\pi = (rv, d, c'')$, the verifier checks whether the depth d is correctly bounded by $\log_k(p_p(|X|, t))$. When $t \le k^C$, d is bounded by a constant, since both t

and $|X|$ are polynomially bounded by k. Thus there must exist a particular constant D, such that the probability that P^* cheats successfully while giving a proof with depth exactly D is polynomial, which violates (2). Therefore, it suffice to show that no prover can prove a false statement by giving a proof with a constant depth.

Assume for contradiction that there exists a deterministic polynomial-time cheating prover P^* a constant D and a polynomial function p such that there exists an infinite sequence of $k \in N$ and $n \in \{0,1\}^*$, such that, P^* succeeds in convincing the honest verifier V of some false statement $X = (M, x, t) \notin \mathcal{N}$ by outputting a proof with $d \leq D$, with probability at least $1/p(k)$. Then we construct another deterministic polynomial time machine B that distinguishes encryption to a random value r and encryption to 0 with non-negligible probability. This violates the semantic security of the fully homomorphic encryption scheme and then the theorem follows.

Fix any k, n for which P^* succeeds in cheating with probability $1/p(k)$. P^*, on input 1^k and $pp = (id, pk, c)$, outputs a statement $X = (M, x, t)$ together with a proof $\pi = (rv, d, c'')$. Consider a wrapper machine A that on input $(1^k, id, z)$, runs $P^*(1^k, (id, z))$ internally, and outputs only the first part rv of P^*'s output. Since the CRHF $\mathcal{H}$ is extractable, it follows from Lemma 2 that there exists a deterministic polynomial time machine E_D, such that, there exists a negligible function μ such that for every $z \in \{0,1\}^*$, except from a $\mu(k)$ fraction of $id \in I_k$, when $A(1^k, id, z)$ outputs a valid hash value rv, $E_D(1^k, id, z)$ outputs a tree $\mathcal{T}$ that is consistent with rv, that is, $\mathsf{merkle}_k(id, D, \mathcal{T}, rv) = 1$. Given E_D, we are now ready to construct machine B.

Machine B on input 1^k and $pp = (id, pk, c)$, internally runs both $P^*(1^k, pp)$ and $E_D(1^k, id, z = (pk, c))$ and outputs both the outputs of P^* and E_D, that is, $(X, \pi = (rv, d, c''))$ and $\mathcal{T}$.

Let r be the value encrypted in c under pk, and $(q_1, \ldots, q_s) = Q(X, r)$ the PCP queries computed from X and r. We show that the probability that $\mathcal{T}$ contains a valid answer for PCP queries $(q_1, \ldots, q_s)$ with respect to X is at least $1/2p(k)$. In other words, machine B is able to prove a false statement "in plaintext" with probability at least $1/2p(k)$. This is proved through the following two Claims, which respectively says that except from negligible probabilities, i) whenever the root value rv output by B is a valid hash value, the tree $\mathcal{T}$ is consistent (w.r.t. merkle) with rv (in Claim 1), and ii) whenever the proof π output by B contains in the encryption c'' an authentication path (a, ap) consistent (w.r.t. verify) with rv, and $\mathcal{T}$ is consistent with rv, then $\mathcal{T}$ contains (a, ap) (in Claim 2).

Claim 1. *There exists a negligible function μ_1, such that for all $k \in N$ and $n \in \{0,1\}^*$,*

$$\Pr\Big[(pp, sp) \leftarrow D(1^k, 1^n),\ (X, \pi, \mathcal{T}) = B(1^k, pp)\ :\ pp = (id, pk, c),\ \pi = (rv, d, c''),\ \mathsf{valid}_k(id, rv) = 1,\ \mathsf{merkle}_k(id, D, rv, \mathcal{T}) \neq 1\Big] \leq \mu_1(k)$$

Claim 2. *There exists a negligible function μ_1, such that for all $k \in N$ and $n \in \{0,1\}^*$,*

$$\Pr\Big[(pp = (id, pk, c), sp = (sk, r)) \leftarrow D(1^k, 1^n),\ (X, \pi = (rv, d, c''), \mathcal{T}) = B(1^k, pp)\ :\ (\vec{a}, \vec{ap}) = \mathrm{Dec}_{sk}(c''),\ \mathsf{merkle}_k(id, D, rv, \mathcal{T}) = 1,\ \exists j \in [|\vec{a}|]\ \mathsf{verify}_k(id, a_j, ap_j) = 1 \textit{ but } \mathcal{T} \textit{ does not contain } a_j, ap_j\Big] \leq \mu_2(k)$$

Claim 1 follows from the fact that by construction, the tree $\mathcal{T}$ output by B is from E_D, which is a good extractor for wrapper A that outputs the root value from P^*. Thus it follows directly from the extractability of the Merkle tree that except from negligible probability whenever rv is valid, $\mathcal{T}$ is consistent. Claim 2 essentially follows from the collision resistence property of the hash

function $\mathcal{H}$. Assume for contradiction that with some polynomial probability, the proof π output by B contains in the encryption c'' an authentication path (a, ap) consistent with rv that does not appear in the tree $\mathcal{T}$. Let q be the leaf node corresponding to (a, ap). Then consider the path (a', ap') from root to leaf q in $\mathcal{T}$. As long as $\mathcal{T}$ is consistent with rv, we must find a collision of the hash function $H_{k,id}$ on (a, ap) and (a', ap'). This violates the collision resistence property of $\mathcal{H}$. Thus Claim 2 holds.

Combining Claim 1 and 2, we have that except from probability $\mu_1(k) + \mu_2(k)$, the output $(X, \pi = (rv, d, c''), \mathcal{T})$ of $B(1^k, pp = (id, pk, c))$ satisfies that when π is a convincing proof for X, the following two conditions hold:

1. rv is valid and thus $\mathcal{T}$ is consistent with rv (by Claim 1).
2. c'' is an encryption to authentication paths (a_j, ap_j) for $j \in [s]$ satisfying that each a_j, ap_j is consistent and $(a_1, \ldots, a_s)$ convinces a PCP verifier with random tape r (where r is the encrypted value in c). Thus $\mathcal{T}$ contains all (a_j, ap_j) (by Claim 2 and the first condition).

Therefore $\mathcal{T}$ contains "in plaintext" the answers that convinces a PCP verifer with random tape r. Then by our hypothesis, B on input 1^k, id, pk and an encryption to r under pk, is able to convince a PCP verifer with random tape r with probability at least $1/2p(k)$. However, when B receives instead (1^k, id, pk and) an encryption to 0 under pk, by the soundness of the PCP system, except from negligible probability, it cannot output answers that convinces a PCP verifier (since the verifier now has completely private random tape). Thus B distinguishes encryption to a random string r and encryption to 0 with probability at least $1/3p(k)$. This violates the semantic security of the fully homomorphic encryption scheme, and gives a contradiction. □

6 Delegation without Rejection Problem

Loosely speaking, a delegation scheme is a 2-stage protocol between a delegator D and a worker W. In an off-line stage, the delegator does some pre-processing based on the function F it wants to compute and generates some public and secret information pp and sp. Later in an on-line stage, the delegator can delegate the computation of F on many inputs to the worker efficiently; the worker evaluates the function and proves to the delegator that the output it returns is correct. The key property of a delegation scheme is that the computation and communication complexity of the delegator in the on-line stage is *polylogarithmic* in the computation complexity of the function. In other words, the delegator can verify the computation done by the untrusted worker much (polylogarithmically) more efficiently than evaluating the function on its own. Recently, Genenaro, Gentry and Parno [GGP10] and Chung, Kalai and Vadhan [CKV10] presented two constructions of delegation schemes, both relying on fully homomorphic encryption schemes. Their schemes, however, share one restriction, known as the *rejection problem*, that is, a cheating worker can break the soundness of the scheme if it learns the verdict of the delegator on whether it accepts or rejects a proof from the worker. Their solution is either to assume that the worker does not learn the verdict or to perform the off-line stage fresh again after every time the delegator rejects a proof. In Section 6.2.2, we construct a two-message delegation scheme that does not suffer the rejection problem. Furthermore, the soundness of previous schemes relies on the use of fully homomorphic encryption schemes, which makes the computational complexity of their scheme depends on the worst case complexity of the function being computed. In Section 6.2.1, we construct a designated verifier delegation scheme, whose soundness relies on the use of designated verifier CS proof and as a result has instance-based complexity for the worker and the delegator in the on-line stage. That is, the complexity of the delegator and the worker in the on-line stage depends on the complexity of the compuation on the particular input being delegated.

6.1 Delegation Schemes

In this section, we formally describe the model of delegation that we consider. Our model is essentially the same as that in [GGP10, CKV10], except from the following three aspects. First, we require that the computational complexity of the worker and the delegator in the on-line stage, computing a function F (represented as a Turing machine M) on input x, to depend solely on the running time t of M on this particular input x, instead of the worst case running time of M; we call this property *instance-based complexity*. Second, we require soundness of the delegation scheme to hold even if a cheating worker learns the verdicts of the delegator in the on-line stage; we call this property *robust soundness*. Third, our scheme only handles delegation of *polynomial-time computation*. The latter seems counter-intuitive: the computational complexity of the delegator (in the on-line stage) is at least polynomial in the length $|x|$ of the input (the delegator needs to at least read the input); if delegation can only handle computation in time polynomial in $|x|$, then what does the delegator gain by engaging in delegation at all? We note that the delegator still gains since the computational complexity of delegation is of a *fixed* polynomial in $|x|$ (and logarithm of the time of the computation), while the complexity of the delegated computation can be of an *arbitrary* polynomial. Furthermore, we remark that this restriction is actually inherent for all delegation schemes whose soundness is based on some polynomial time hardness assumptions. The reason is that since soundness only holds against computationally bounded workers (otherwise a cheating worker can break the polynomial time hardness assumption that soundness is based on), then the worker, being computationally bounded, can only compute polynomial time computable functions for the delegator. (We note that this restriction also applies to previous constructions [GGP10, CKV10], although it is not stated explicitly.)

Definition 9 (Delegation with Robust Soundness)**.** *A delegation scheme is an interactive protocol* $\mathsf{Del} = (D = (A_1, A_2), W)$ *consisting of the following two stages:*

Off-Line Stage: *The delegator A_1 on input a security parameter k and a* polynomial-time computable *function $F : \{0,1\}^n \to \{0,1\}^m$, represented as a Turing machine M and a polynomial time bound $T = k^c$ for M, outputs a public string pp and a private string sp, that is, $(pp, sp) = A_1(1^k, F)$. We will use the notation M, n, m, and T as the Turing machine and parameters associated with F below.*

On-Line Stage: *The delegator A_2 and the worker W on input 1^k, F and an input $x \in \{0,1\}^n$, interacts in an interactive protocol. At the end of the protocol, the delegator A_2 decides to accept or reject; if it accepts, then it additionally outputs a value y.*

Furthermore, Del *satisfies the following properties.*

Efficiency: *The computational complexity of worker W and the delegator A_2 in the on-line stage is bounded by* $\mathrm{poly}(k, T)$ *and* $\mathrm{poly}(k, n, m, \log T)$ *respectively. The total communication complexity of the on-line and off-line stage is bounded by* $\mathrm{poly}(k, \log n, m, \log T)$.

Furthermore if the computational complexity of W and A_2 depends on the running time t of M on input x (i.e., resepctively $\mathrm{poly}(k, t)$ *and* $\mathrm{poly}(k, n, m, \log t)$*), instead of the worst case running time T of M, we say that the delegation scheme has* instance-basec complexity.

Completeness: *For all k, n, m, T, F and x, after running the delegation scheme (D, W) on these inputs as specified above, A_2 accepts with probability 1.*

Robust Soundness: *For every cheating $\mathcal{PPT}$ worker W^*, every constant C, there exists a negligible function μ, such that for every $k \in N$, every $F = (M, n, m, T)$ with $T \leq k^C$, the probability that W^* wins in the following security game $Sec(W^*, 1^k, F)$ is bounded by $\mu(k)$.*

1. *The delegator A_1 executes the off-line stage by generating $(pp, sp) \leftarrow A_1(1^k, F)$.*
2. *The cheating worker W^* interacts with A_2 in an arbitrary number of iterations of its choice. In the i^{th} iteration, W^* selects an input $x_i \in \{0,1\}^n$ and interacts with A_2 on input $(1^k, F, x_i, pp, sp)$; after A_2 outputs a verdict b_i, W^* learns b_i. If A_2 accepts in round i, that is $b_i = 1$, let y_i be value A_2 outputs at the end of the protocol.*

W^* wins in the game, if there exists an i in which A_2 accepts (i.e., $b_i = 1$) but $y_i \neq F(x_i)$.

In the above defined soundness game *Sec*, the cheating worker learns the verdict of the delegator immediately after each execution of the on-line stage. In contrast, previous works [GGP10, CKV10] consider a soundness game that terminates once the delegator rejects a proof from the cheating worker; we call delegation scheme with such soundness guarantee a delegation scheme without verification oracle.

In this work, we will also consider the following additional properties. We say that a delegation scheme is a one-message delegation scheme, if the on-line stage contains only a single message from the worker to the delegator (that is, A_2 does not send any message). We say that a delegation scheme is on-line if the off-line stage is empty. Furthermore, we say that a delegation scheme has an efficient off-line stage if the computational complexity of the delegator A_1 in the off-line stage is $\text{poly}(k, n, m, |M|, \log T)$.

Remark 3. *In the above definition, the verifier runs in time polynomial in the length n of the input x (beyond running in time polylogarithmic in the running time of the computation T). This seems necessarily, since the verifier needs to at least read the entire input x. However, if one is willing to assumes that the input x is given in an error correcting code (as is done in [BFLS91] and some follow-up works), it would be possible to achieve verifier's computational complexity that is poly-logarithmic in the length of x. Such a delegation scheme can be constructed using essentially the same method below from a designated verifier CS proof that has verifier's complexity of polylogarithmic in the length of the input, which in turn can be constructed almost identically as in Section 5 but relying on a PCP of proximity system for the non-deterministic CS language $\mathcal{N}$ as in [CKLR11].*

6.2 Our Constructions

We present two delegation schemes. The first scheme $\mathsf{Del}_1 = (W_1, D_1)$, as previous constructions, only satisfies soundness without verification oracle; but it improves previous works on that it has an efficient off-line stage, a one-message on-line stage, and instance-based complexity. The second scheme $\mathsf{Del}_2 = (W_2, D_2)$ satisfies soundness with verification oracle and is further completely on-line.

6.2.1 Designated Verifier CS Proofs for CS Language $\mathcal{L}$

Our delegation schemes will rely on a designated verifier CS proof system for the (deterministic) CS language $\mathcal{L}$, which includes all quadruples $X = (M, x, y, t)$ such that $M(x) = y$ in t steps. (See Definition 5 for a formal definition.) In section 5.2, we constructed a designated verifier CS proof system (D_N, P_N, V_N) for the non-deterministic CS language $\mathcal{N}$. We show how to transform that scheme to a designated verifier CS proof system (D_L, P_L, V_L) for $\mathcal{L}$.

Towards this, we first show that every statement $X = (M, x, y, t) \in \mathcal{L}$ can be converted efficiently into a statement $X' \in \mathcal{N}$ together with a corresponding witness w. The conversion algorithm $\mathsf{convert}(X)$ proceeds as follows:

$\mathsf{convert}(X = (M, x, y, t))$**:** Run M on input x, producing the output y and a t-step history σ of a computation of M outputting y on input x. Consider the following relation R.

$R(X, \sigma) = 1$ *if and only if* σ *is the* t*-step history of a computation of* M *outputting* y *on input* x*.*

Evaluates $R((M, x, y, t), \sigma)$ and records the number of steps t' taken by R. Then set the statement $X' \in \mathcal{N}$ to $(R, (M, x, y, t), t')$ and the witness $w = \sigma$.

Notice that R is polynomial time computable. Thus $t' = \text{poly}(|X|, |\sigma|) = \text{poly}(|X|, t)$ bounded by t^β for some constant β. Therefore, the length of the new statement $|X'|$ is bounded by $|X|^\gamma$ for some constant γ, and the computational complexity of $\mathsf{convert}$ is $\text{poly}(|X|, t)$. Then the designated verifier CS proof system (D_L, P_L, V_L) for $\mathcal{L}$ proceeds as follows:

Set-up: D_L, on input 1^k and $1^{|X|}$, sets $n = |X|^\gamma$ and runs $D_N(1^k, 1^n)$ producing (pp, sp).

Prover's Message: P_L on input 1^k, $X = (M, x, y, t)$, and pp, converts X into a statement $X' \in \mathcal{N}$ and its witness w using the conversion algorithm $\mathsf{convert}$, runs the honest prover strategy $P_N(1^k, pp, X', w)$ to generate a proof π' and outputs $\pi = (X', \pi)$.

Verifier's Decision: V_L on input 1^k, $X = (M, x, y, t)$, pp, sp and $\pi = (X', \pi')$, first checks whether X' is well formed and consists of R, (M, x, y, t) and t'; furthermore, it checks whether $t' \leq t^\beta$. It aborts if any of the conditions does not hold. Otherwise, it runs the honest verifies strategy $V_N(1^k, pp, sp, X', \pi')$ and accepts if and only if V_N accepts.

(D_L, P_L, V_L) is a designated verifier CS proof system for $\mathcal{L}$. It is easy to see that the computational complexity of the prover is $\text{poly}(k, |X|, t)$ and the length of the proof is $\text{poly}(k, |X|)$. Completeness follows from the completeness of (D_N, P_N, V_N). Furthermore, weak soundness follows from the weak soundness of (D_N, P_N, V_N) and the fact that V_L checks that the time t' sent by P_L is correctly bounded by t^β; this is because if a cheating prover P_L^* is able to prove a false statement X with $t \leq k^C$ for some C, it must prove a false statement $X' \in \mathcal{L}$ with some $t' \leq t^\beta \leq k^{C\beta}$ using (D_N, P_N, V_N). This violates the weak soundness of (D_N, P_N, V_N).

6.2.2 One-Message Delegation with Instance-based Complexity

Let (D, P, V) be a designated verifier CS proof system for the non-deterministic CS language $\mathcal{L}$. The delegation scheme (D_1, W_1) proceeds as follows:

Off-Line Stage: The delegator D_1, on input 1^k and a function $F = (M, n, m, T)$, runs the set-up algorithm D on input $(1^k, 1^l)$ with $l = |M| + n + m + |T|$, producing (pp, sp).

On-Line Stage: The worker W_1 on input 1^k, F, pp and an input $x \in \{0,1\}^n$, first evaluates M on input x to obtain the output $y = M(x)$ and measure the time t taken by the computation. It then runs the honest prover strategy $P(1^k, pp, X = (M, x, y, t))$ to generate a proof π. Finally, W_1 outputs $y, (t, \pi)$.

The delegator D_1 after receiving y and (t, π), checks whether $t \leq T$. It aborts if $t > T$; otherwise, it runs the honest verifier strategy $V(1^k, pp, sp, X, \pi)$ to verify the proof, and accepts if and only if V accepts.

The completeness of Del_1 follows directly from that of the designated verifier CS proof. Furthermore, since the designated verifier CS proof system has instance based complexity. That is, the computational complexity of the prover and verifier on input $X = (M, x, y, t)$ are respectively

$\text{poly}(k,t)$ and $\text{poly}(k,|X|,t) = \text{poly}(k,|x|,|y|,\log t)$, and the total length of the public information and the CS proof is $\text{poly}(k,\log|X|,\log t) = \text{poly}(k,\log|x|,\log|y|,\log t)$ (See Remark 2), all independent the worst case running time of M. The delegation scheme Del_1 derived from it also has instance based complexity. More precisely, in the on-line stage, the deletagor and the worker on input F and x runs in time $\text{poly}(k,n,m,\log t)$ and $\text{poly}(k,t)$, where t is the running time of M on input x; in the off-line stage, the delegator runs in time $\text{poly}(k,n,m,\log T)$, where T is the worst case running time of M[5]. Furthermore, the total communication complexity including the public key and the on-line proof is $\text{poly}(k,\log n,m,\log T)$. The soundness of the scheme follows from the weak soundness of the designated verifier CS proof. However, since the latter only holds against a cheating prover who does not see the verdict of the verifier. Scheme Del_1 only achieves soundness without verification oracle. Finally, we remark that Del_1 has an efficient off-line stage with computational complexity $\text{poly}(k,\log n,m,\log T)$ independent of the complexity of the compuation, and the on-line stage contains only a single message from the worker to the delegator.

6.2.3 On-Line Delegation Satisfying Soundness with Verification Oracle

Soundness without verification oracle means that every time after the delegator rejects a proof, it needs to execute the off-line stage fresh again to prevent a cheating worker (aware of its decision) from cheating in later proofs. Although the delegation scheme Del_1 satisfies only soundness without verification oracle, it has a very efficient off-line stage, with computational complexity $\text{poly}(k,n,m,\log T)$. Therefore, the delegator can, in fact, afford to run the off-line stage every time before delegating a computation. By doing so, soundness holds even if the cheating worker learns the delegator's decisions, yielding a two-message delegation scheme satisfying robust soundness.

Off-Line Stage: Empty.

On-Line Stage: The delegator D_2, on input 1^k and a function $F = (M,n,m,T)$, runs $D_1(1^k,F)$ in off-line stage to generate (pp,sp); it sends pp to the worker.

The worker W_2 on input 1^k, F, pp and an input $x \in \{0,1\}^n$, sends back $(y,\pi) = W_1(1^k,pp,F,x)$.

D_2 reaches a decision exactly as D_1 does.

References

[AF07] Masayuki Abe and Serge Fehr. Perfect nizk with adaptive soundness. In *TCC*, pages 118–136, 2007.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *STOC*, pages 21–31, 1991.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM J. Comput.*, 38(5):1661–1694, 2008.

[BP04] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In *ASIACRYPT*, pages 48–62, 2004.

[5] Recall that the delegator used T as an upper bound on the running time to generate the off-line message

[BSGH+05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Short pcps verifiable in polylogarithmic time. In *IEEE Conference on Computational Complexity*, pages 120–134, 2005.

[BSGH+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust pcps of proximity, shorter pcps, and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006.

[BV11] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. To appear in TCC 2010, 2011.

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Extractable perfectly one-way functions. In *ICALP (2)*, pages 449–460, 2008.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[CKLR11] Kai-Min Chung, Yael Tauman Kalai, Feng-Hao Liu, and Ran Raz. Memory delegation. Cryptology ePrint Archive, Report 2011/273, 2011. `http://eprint.iacr.org/`.

[CKV10] Kai-Min Chung, Yael Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In *CRYPTO*, pages 483–501, 2010.

[CL08] Giovanni Di Crescenzo and Helger Lipmaa. Succinct np proofs from an extractability assumption. In *CiE*, pages 175–185, 2008.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT*, pages 402–414, 1999.

[Dam91] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO*, pages 445–456, 1991.

[Din07] Irit Dinur. The pcp theorem by gap amplification. *J. ACM*, 54(3):12, 2007.

[DNRS03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. *J. ACM*, 50(6):852–921, 2003.

[FL93] Lance Fortnow and Carsten Lund. Interactive proof systems and alternating time-space complexity. *Theor. Comput. Sci.*, 113(1):55–73, 1993.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[Gen11] Craig Gentry. Fully homomorphic encryption without bootstrapping. Manuscript, 2011.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, pages 465–482, 2010.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *FOCS*, pages 102–, 2003.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122, 2008.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[Gol01] Oded Goldreich. *Foundations of Cryptography — Basic Tools*. Cambridge University Press, 2001.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, pages 321–340, 2010.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732, 1992.

[Kil95] Joe Kilian. Improved efficient arguments (preliminary version). In *CRYPTO*, pages 311–324, 1995.

[KR08] Yael Tauman Kalai and Ran Raz. Interactive pcp. In *ICALP (2)*, pages 536–547, 2008.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[Mer89] Ralph C. Merkle. A certified digital signature. In *CRYPTO*, pages 218–238, 1989.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

[PS94] Alexander Polishchuk and Daniel A. Spielman. Nearly-linear size holographic proofs. In *STOC*, pages 194–203, 1994.

[Sha92] Adi Shamir. Ip = pspace. *J. ACM*, 39(4):869–877, 1992.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43, 2010.

A General Definitions

A.1 Witness Relations

We recall the definition of a witness relation for a $\mathcal{NP}$ language [Gol01].

Definition 10 (Witness relation)**.** *A* witness relation *for a language $L \in \mathcal{NEXP}$ is a binary relation R_L that characterizes L by $L = \{x : \exists y \, s.t. \, (x, y) \in R_L\}$*

We say that y is a witness for the membership $x \in L$ if $(x, y) \in R_L$.

A.2 Indistinguishability

Definition 11 (Computational Indistinguishability)**.** *Let Y be a countable set. Two ensembles $\{A_{k,y}\}_{k\in N,y\in Y}$ and $\{B_{k,y}\}_{k\in N,y\in Y}$ are said to be* computationally indistinguishable *(denoted by $\{A_{k,y}\}_{k\in N,y\in Y} \approx \{B_{k,y}\}_{k\in N,y\in Y}$), if for every $\mathcal{PPT}$ "distinguishing" machine D, there exists a negligible function $\nu(\cdot)$ so that for every $k \in N, y \in Y$:*

$$\left|\Pr\left[a \leftarrow A_{k,y} \ : \ D(1^k, y, a) = 1\right] - \Pr\left[b \leftarrow B_{k,y} \ : \ D(1^k, y, b) = 1\right]\right| < \nu(k)$$

A.3 Interactive Proofs

We use the standard definitions of interactive proofs (and interactive Turing machines) [GMR89] and arguments (a.k.a. computationally-sound proofs) [BCC88]. Given a pair of interactive Turing machines, P and V, we denote by $\langle P(w), V\rangle(x)$ the random variable representing the (local) output of V, on common input x, when interacting with machine P with private input w, when the random input to each machine is uniformly and independently chosen.

Definition 12 (Interactive Proof System)**.** *A pair of interactive machines $\langle P, V\rangle$ is called an interactive proof system for a language L if there is a negligible function $\nu(\cdot)$ such that the following two conditions hold :*

- Completeness: *For every $x \in L$, and every $w \in R_L(x)$,* $\Pr\left[\langle P(w), V\rangle(x) = 1\right] = 1$
- Soundness: *For every $x \in \{0,1\}^n - L$, and every interactive machine B,* $\Pr\left[\langle B, V\rangle(x) = 1\right] \leq \nu(n)$

In case that the soundness condition is required to hold only with respect to a computationally bounded prover, the pair $\langle P, V\rangle$ is called an interactive argument *system.*