6.892 Computing on Encrypted Data October 07, 2013

Lecture 5

Lecturer: Yael Kalai *Scribe: Justin Holmgren*

1 Introduction

So far, we have studied the problem of outsourcing computation when we want our data to be private from the server. Today, we will forget about privacy and focus on the problem of verifying that the server computes correctly for us. Specifically, we will look at decision problems. We consider the problem where we send the cloud x, and the cloud (efficiently) computes both $f(x)$ as well as a proof of correctness. This is called delegation [2].

Definition 1. *A (statistical) delegation protocol for a language L which is computable in time T is a protocol between a prover P and verifier V satisfying the following properties.*

1. *(Completeness) If x is in the language, the prover always succeeds.*

$$\forall x \in L, Pr[(P,V)(x) = 1] = 1$$

2. *(Soundness) If x is not in the language, the prover almost certainly fails. For computational delegation protocols, this only must be true for computationally bounded provers.*

$$\forall x \notin L, \forall P^*, Pr[(P^*,V)(x) = 1] = \text{negl}(|x|)$$

3. *(Efficiency) The prover must run in time* $\text{poly}(T)$*, and the verifier must run in time much less than T (we will show a runnning time of* $\text{polylog}(T) \cdot n$*).*

An example of a language L is the set of all tuples $(x, f(x))$ for some polynomial-time computable function f. Thus if the prover proves membership in L, he is proving that $f(x)$ was computed correctly.

2 What is known

There has been a lot of work on computational delegation, but much less work has been done on statistical delegation.

2.1 Statistical Delegation

One result that seems relevant to statistical delegation is the theorem that $IP = PSPACE$ [5, 7]. This theorem shows that if a language is computable in space S, a prover can prove to a verifier that x is in the language, satisfying the completeness and soundness conditions given above. However, the efficiency conditions are not satisfied. The verifier runs in time $\text{poly}(S) \cdot n$, and the prover runs in time $2^{\Theta(S^2)}$.

In particular, if the language is decidable in $\Omega(n)$ space and $2^{O(n)}$ time, then the prover is too slow.

Note: if in our definition of delegation scheme efficiency, we were concerned with space usage rather than time usage, the $IP = PSPACE$ construction would constitute an efficient delegation scheme.

Open Problem 1. *Construct a statistical delegation scheme for languages (decidable in time T and space S) such that the prover runs in time* $\text{poly}(T)$ *and the verifier runs in time* $\text{poly}(S) \cdot n$*.*

This is best possible.

2.2 Computational Delegation

Computational delegation is a strictly easier problem than statistical delegation, and there are fantastic results for it.

To construct a computational delegation scheme, we will make use of the PCP theorem [1]. We show a 4 round protocol in which the prover runs in time $\text{poly}(T)$, the verifier runs in time $\text{polylog}(T) \cdot n$, and the communication complexity is $\text{polylog}(T)$, assuming the existence of collision-resistant hash functions.

The question of whether this can be improved to a 2 round protocol has been an area of intense interest. V's first message to P is typically independent of x, so P's response in a 2-round protocol would be a certification of x's membership in L, which is appealing.

2-round protocols are known to exist, but only under strong and non-standard cryptographic assumptions (like the random oracle model).

2.2.1 Kilian-Micali Protocol

The Kilian-Micali delegation protocol [6, 4] makes use of probabilistically checkable proofs (PCP). Essentially, given any language L decidable in time T and any x in L, a prover can construct a proof π_x of x's membership in L which is of length $\text{poly}(T)$ (and also takes time $\text{poly}(T)$ to compute π_x. The verifier is given oracle access to bits of π, and in $\text{polylog}(T)$ time, it can decide whether x is in L with perfect completeness and negligible soundness error. To be precise,

Theorem 2. *For any language $L \in \mathbf{NP}$, there exists a verifier V making $O(1)$ oracle queries using $O(\log n)$ randomness such that:*

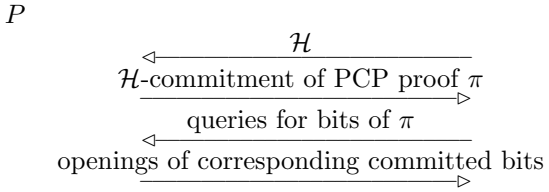
1. $\forall x \in L \exists \pi : Pr[V^\pi(x) = 1] = 1$
2. $\forall x \notin L \forall \pi : Pr[V^\pi(x) = 1] \leq \frac{1}{2}$

Furthermore, when x is in L and L is decidable in time T, π is computable in time $\text{poly}(T)$.

This is not yet a delegation scheme because V's oracle access to π_x is outside our model. In particular, V receiving the entirety of π_x would require time $\text{poly}(T)$, which is too much. P's commitment to the entirety of π_x is essential - if V were to make queries to P which could be answered arbitrarily, then soundness would be entirely lost. What we need is a way for P to commit to $\text{poly}(T)$ bits in a $\text{polylog}(T)$-bit message, and a way for P to open $\text{polylog}(T)$ bits of V's choosing in a $\text{polylog}(T)$-bit message. We do not need the hiding properties of the commitment; we only require that the commitment be binding.

Merkle hashes, also known as tree commitments, provide a way to achieve this. The idea is that a collision-resistant hash function automatically provides a binding commitment, and that an n-bit message can be hashed in a tree structure such that only $O(\log(n))$ bits need be revealed to open the commitment (see Figure 1).

In more detail, assume we have a family of collision-resistant hash functions $\mathcal{H} = \{\mathcal{H}_k : \{0,1\}^* \to \{0,1\}^k\}$. That is, for any PPT adversary $\mathcal{A}$, the probability over hash functions h in $\mathcal{H}_k$ that $\mathcal{A}$ outputs distinct x_1 and x_2 with $h(x_1) = h(x_2)$ is negligible in k. If we split our input into blocks of size k, and repeatedly hash adjacent blocks into a single k-bit block, we have a tree of depth $\log_k(n)$, where a node is the hash of the concatenation of its children. The root of this tree together with the depth comprises a commitment of all the leaf nodes. In order to properly utilize the collision resistance properties of our hash function family, we let the verifier first send a description of the hash function to use to the prover.



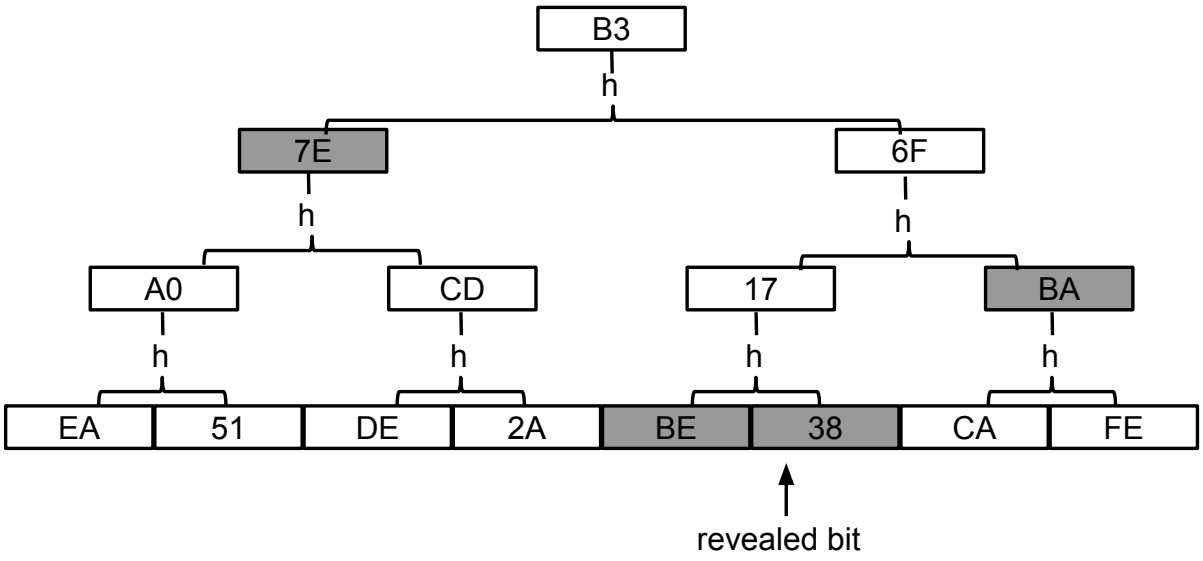


Figure 1: A Merkle hash tree

It is easy to see that any adversary which can open any bit in two different ways must have found a hash collision at one of the levels of the tree (see Figure 2), so this is impossible assuming the collision-resistance of the hash family.

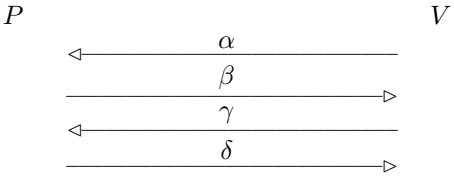
When we use this commitment scheme in our template for a delegation scheme, completeness is easy to see. Efficiency also follows easily. We will prove computational soundness by demonstrating that if an adversary breaks the soundness of our delegation scheme, this adversary violates the PCP theorem.

Suppose there is a cheating prover P^* such that for some x not in L, $(P^*, V)(x) = 1$ with non-negligible probability ϵ. Recall that P^*'s first message must be a tuple of a root node together with a depth. If P^* has non-negligible probability of successfully cheating, then a non-negligible fraction of the time he must produce a (root, depth) tuple for which he has a non-negligible probability of satisfying the verifier's challenges. So if we run the verifier polynomially many times (in n and $1/\epsilon$, rewinding the prover each time), either P^* will open a bit in two different ways, or the prover will have demonstrated a false proof for which a constant fraction of the bits are convincing, thus violating the PCP theorem.

2.2.2 Reducing Round Complexity

If we assume the random oracle model, we can use the Fiat-Shamir heuristic to transform this protocol (or any public-coin protocol) into a 2-round protocol. Essentially all but the first of the verifier's messages are computed by the prover, using "randomness" which is computed as a random function of all previous messages.

Given a protocol of the form



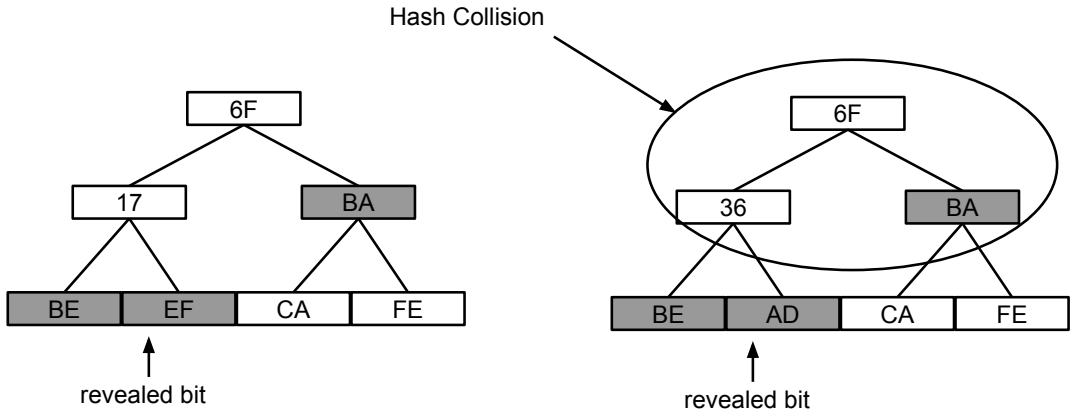


Figure 2: Opening a committed bit in two different ways requires finding a hash collision

P V

We transform this into the protocol

α, H ←

→ β, γ, δ

γ is generated using randomness $H(\alpha|\beta)$

The verifier accepts if $\alpha, \beta, \gamma, \delta$ would have been an accepting transcript in the original protocol.

2.3 Public-coin Statistical Delegation to Computational Delegation

We now show another way to get a two-round computational delegation scheme, assuming the existence of a public-coin statistical delegation scheme and fully homomorphic encryption which is secure against all subexponential-time adversaries. Here for simplicity we will assume that the language of our delegation scheme is polynomial time. If the language is not polynomial time, then some security parameters need to be appropriately larger (e.g. the security parameter for FHE).

Suppose we are given a public-coin statistical delegation protocol in which each of the verifier's messages are chosen independently of any previous prover messages.

P V

v_1 ←

→ p_1

$\vdots$

v_n ←

→ p_n

We construct the following two-round computational delegation scheme.

P' V'

$v_1, \mathsf{Enc}_{pk_2}(v_2), \ldots, \mathsf{Enc}_{pk_n}(v_2|\cdots|v_n)$ ←

→ $p_1, \mathsf{Enc}_{pk_2}(p_2), \ldots, \mathsf{Enc}_{pk_n}(p_n)$

In the first round, the verifier sends an encryption of each of the messages he would have sent (he can compute this ahead of time because none of the messages depend on what the prover sends). v_i is encrypted under keys $pk_i \ldots pk_n$, and each of the provers outputs p_i is encrypted under pk_i. Intuitively, this ensures

that the prover cannot use the value of v_i to influence p_j for any $j < i$.

Efficiency and completeness easily carry over from the statistical delegation scheme. The interesting property to prove is soundness. We reduce the soundness of (P', V') to the IND-CPA security of the FHE cryptosystem against subexponential time adversaries. We only show the case where $n = 2$.

Suppose there is a cheating prover P^* and an element x not in L such that $\Pr[(P^*, V')(x) = 1] \geq \epsilon$ for some non-negligible ϵ. Define the predicate $\text{valid}(v_1, p_1, v_2, p_2)$ to be true if the corresponding transcript in (P, V) would cause the verifier to accept. This is efficiently computable because (P, V) is a public-coin protocol. Define the adversary $\mathcal{A}$ to break FHE as follows. First, $\mathcal{A}$ independently chooses a random verifier message v_1 and two random verifier messages v_2^0 and v_2^1 and outputs these as the two plaintexts it would like to distinguish. $\mathcal{A}$ then receives some ciphertext C where $C = \mathsf{Enc}_{pk}(v_2^b)$ for $b \in \{0, 1\}$, and passes (v_1, C) to P^* as a simulated message from V'. Then with some non-negligible probability ϵ, P^* computes p_1 and $\mathsf{Enc}(p_2)$ such that $\text{valid}(v_1, p_1, v_2^b, p_2)$. In particular, $\Pr\left[\exists p_2 : \text{valid}(v_1, p_1, v_2^b, p_2)\right] \geq \epsilon$.

However, by the statistical soundness of (P, V), since x is not in L, $\forall v_1, p_1$, there are negligibly many v_2 such that $\exists p_2 : \text{valid}(v_1, p_1, v_2, p_2)$. This means that intuitively P^* must have extracted some information from $\mathsf{Enc}_{pk}(v_2^b)$ to compute p_1, but this is not really a formal statement. Because v_2^{1-b} was chosen uniformly at random independently of inputs to P^*, there is negligible probability that $\exists p_2 : \text{valid}(v_1, p_1, v_2^{1-b}, p_2)$. When compared with the conclusion of the last paragraph, this gives a non-negligible advantage in distinguishing which ciphertext we received. $\mathcal{A}$ checks all possibilities for p_2, and checks whether $\text{valid}(v_1, p_1, v_2^0, p_2)$ is ever true. If it is, then $\mathcal{A}$ returns v_2^0. Otherwise, it returns v_2^1.

Because of the requirements on verifier efficiency, p_2 must have length $o(n)$, so the running time of alA is subexponential - $2^{o(n)}$. So we have reduced the soundness of our constructed computational delegation scheme to the IND-CPA security of our FHE scheme against subexponential adversaries.

Note that this transformation only works on public-coin, history-ignoring protocols. We do know how to transform IP into a public-coin, history-ignoring protocol [3], but this causes a large increase in prover running time.

Open Problem 1. *Is there a prover efficiency preserving mechanism for transforming IP into a public-coin, history-ignoring protocol?*

References

[1] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998.

[2] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 113–122. ACM, 2008.

[3] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 59–68. ACM, 1986.

[4] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732. ACM, 1992.

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

[6] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[7] Adi Shamir. Ip = pspace. *Journal of the ACM (JACM)*, 39(4):869–877, 1992.