COSC 544 Probabilistic Proof Systems 8/31/17

Introductory Lecture

Lecturer: Justin Thaler

1 What This Course Is About

This informal lecture gives a high level overview of the content that will be covered in this course.

1.1 Mathematical Proofs

This course is about different notions of *mathematical proofs*, and their applications in computer science and cryptography. Informally, what we mean by a proof is anything that convinces someone that a statement is true, and a "proof system" is any procedure that decides what is and is not a convincing proof. That is, a proof system is specified by a verification procedure that takes as input any statement and a claimed "proof" that the statement is true, and decides whether or not the proof is valid.

What properties do we want in a proof system? Here are four obvious ones.

- Any true statement should have a convincing proof of its validity. This property is typically referred to as *completeness*.
- No false statement should have a convincing proof. This property is typically referred to as *soundness*.
- Ideally, the verification procedure will be "efficient". Roughly, this means that simple statements should have short (convincing) proofs that can be *checked* quickly.
- Ideally, proving should be efficient too. Roughly, this means that simple statements should have short (convincing) proofs that can be *found* quickly.

Traditionally, a mathematical proof is something that can be written and checked line-by-line for correctness. This traditional notion of proof is precisely the one captured by the complexity class **NP**.

However, over the last 30+ years, computer scientists have studied much more general and exotic notions of proofs. That has transformed computer scientists' notions of what it means to prove something, and has led to major advances in complexity theory and cryptography.

1.2 What kinds of non-traditional proofs will we study?

All of the notions of proofs that we study in this course will be probabilistic in nature. This means that the verification procedure will make random choices, and the soundness guarantee will hold with (very) high probability over those random choices. That is, there will be a (very) small probability that the verification procedure will declare a false statement to be true.

By far, our main focus in this course will be on three types of proof systems: interactive proofs, arguments, and their zero-knowledge variants.

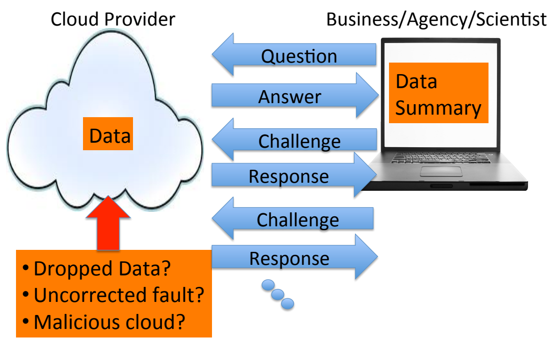

Figure 1: Depiction of an interactive proof used to check that a cloud computing provider is storing and processing a user's data correctly.

1.2.1 Interactive Proofs (IPs)

To understand what an interactive proof is, it is helpful to think of the following application. Imagine a business (verifier) that is using a commercial cloud computing provider to store and process its data. The business sends all of its data up to the cloud (prover), which stores it, while the business stores only a very small "secret" summary of the data. Later, the business asks the cloud a question about its data, typically in the form of a computer program f that the business wants the cloud to run on its data using the cloud's vast computing infrastructure. The cloud does so, and sends the user the claimed output of the program, $f(\mathtt{data})$. Rather than blindly trust that the cloud executed the program on the data correctly, the business can use an interactive proof system (IP) to obtain a formal *guarantee* that the claimed output is correct.

In the IP, the business interrogates the cloud, sending a sequence of challenges, and receiving a sequence of responses. At the end of the interrogation, the business must decide whether to accept the answer as valid or reject it as invalid. See Figure 1 for a diagram of this interaction.

Completeness of the IP means that if the cloud correctly runs the program on the data and follows the prescribed protocol, then the user will be convinced to accept the answer as valid. Soundness of the IP means that if the cloud returns the wrong output, then the user will reject the answer as invalid with high probability *no matter how hard the cloud works to trick the user* into accepting the answer as valid. Intuitively, the interactive nature of the IP lets the business exploit the element of surprise (i.e., the fact that the cloud cannot predict the business' next challenge) to catch a lying cloud in a lie.

Interactive proofs were introduced by Goldwasser, Micali, and Rackoff [GMR89] and Babai [Bab85] in 1985.

It is worth remarking on an interesting difference between IPs and traditional static proofs. Static proofs are *transferrable*, meaning that if Peggy hands Victor a proof that a statement is true, Victor can turn around and convince Charlie that the same statement is true, simply by copying the proof. In contrast, an interactive proof may not be transferrable. Victor can try to convince Charlie that the statement is true by sending Charlie a transcript of his interaction with Peggy, but Charlie will not be convinced unless Charlie trusts that Victor correctly represented the interaction. This is because soundness of the IP only holds if, every time Peggy sends a response to Victor, Peggy does not know with what challenge Victor will respond with next. The transcript alone does not give Charlie a guarantee that this holds.

1.2.2 Argument Systems

Argument systems are IPs, but where the soundness guarantee need only hold against cheating provers that run in polynomial time. Argument systems typically make use of cryptography (roughly speaking, a cheating prover would have to break the cryptosystem to trick the verifier into accepting a false statement, and this is assumed to require superpolynomial time). Argument systems were introduced by Brassard, Chaum, and Crépeau in 1986 [BCC88].

1.2.3 Zero-Knowledge Proofs and Arguments

A zero-knowledge proof or argument is one that reveals nothing to the verifier other than the validity of the statement being proven. Such proof systems have many cryptographic applications. One quintessential example is in authentication.

Suppose that Alice chooses a random password x and publishes a hash $z = h(x)$, where h is a one-way function (this means that given $z = h(x)$ for a randomly chosen x, it should require enormous computational power to find a preimage of z under h (i.e., an x' such that $h(x') = z$). Later, suppose that Alice wants to convince Bob that she is the same person who published z. She can do this by proving to Bob that she knows an x' such that $h(x') = z$. This will convince Bob that Alice is the same person who published z, since it means that either Alice knew x to begin with, or she inverted h (which is assumed to be hard).

How can Alice convince Bob that she knows a preimage of z under h? A trivial proof is for Alice to simply send x to Bob, and Bob can easily check that $h(x) = z$. But this reveals much more information than that Alice knows a preimage of z. In particular, it reveals the preimage itself! And Bob can then impersonate Alice forevermore, since now he too knows the preimage of z.

In order to prevent Bob from learning information that can compromise the password x, it is important that the proof reveal nothing beyond its own validity. This is exactly what a zero-knowledge proof or argument guarantees.

1.2.4 Multi-Prover Interactive Proofs, Probabilistically Checkable Proofs, etc.

In addition to IPs, arguments, and their zero-knowledge variants, we will also study other types of proof systems, especially multi-prover interactive proofs (MIPs), and probabilistically checkable proofs (PCPs).

A MIP is like an IP, except that there are multiple provers, and these provers are assumed not to share information with each other regarding what challenges they receive from the verifier (a common analogy is to placing two or more criminal suspects in separate rooms before interrogating them, to see if they can keep their story straight).

In a PCP the proof is static, as in a traditional mathematical proof, but the verifier is only allowed to read a small number of (randomly chosen) characters from the proof.

MIPs and PCPs are not directly applicable in most cryptographic settings, because they make unrealistic or onerous assumptions about the prover(s): soundness of MIP only holds if there is more than one prover, and these provers do not share information with each other regarding what challenges they receive from the verifier. Similarly, although the verifier only reads a few characters of a PCP, a direct implementation of a PCP would require the prover to transmit the whole proof to the verifier, and this would be the dominant cost in most real-world scenarios.

The main reason we will study MIPs and PCPs is because they are useful in building arguments. Specifically, argument systems are often developed in a two-step process. First, an information-theoretically secure protocol, such as a MIP, PCP, or related construct, is developed for a model involving one or more provers that are assumed to behave in some restricted manner (e.g., in a MIP, the provers are assumed not to send information to each other about the challenges they receive from the verifier). Second, the information-theoretically secure protocol is combined with cryptography to "force" a (single) prover to behave in the restricted manner.

1.3 Context For This Course

In the mid-1980s and 1990s, theoretical computer scientists showed that IPs and arguments can be vastly more efficient (at least, in an asymptotic sense) than traditional static proofs. The foundational results characterizing the power of these protocols (such as **IP=PSPACE** [LFKN92, Sha90], **MIP=NEXP** [BFL91], and the PCP theorem [ALM$^+$98, AS98]), are some of the most influential and celebrated in all of computational complexity theory. But despite their remarkable asymptotic efficiency, these protocols were thought to be wildly impractical, and with good reason: naive implementations of the theory would have had comically high concrete costs (trillions of years for the prover, even for very short computations).

But the last 6 years have seen major improvements in the costs of these proof systems, and they've begun to be implemented, and even used in commercial settings. These works have also generated insights that offer a substantially more detailed understanding of the power of IPs and arguments.

These lecture notes survey the algorithmic insights that underly the foundational results as well as the state of the art proof systems that have been developed in the last 6 years. My goal is to convey these insights in a concise and unified manner, and I hope that these notes can function as a single reference for the state of the art in IPs, argument systems, and their zero-knowledge variants.[1] Open questions are highlighted whenever possible.

[1]There is also a rich literature that seeks to develop MIPs and PCPs for use in establishing hardness of approximation. The focus in these works is on developing PCPs and MIPs with various structural properties that enable efficient reductions from approximation problems such as vertex cover, set cover, etc. For example, Khot's famous Unique Games conjecture is a statement about two-prover MIPs in which the verifier's acceptance criterion satisfies a certain "bijection" property. These structural properties are not relevant in to us in this course, as we are not focused on hardness of approximation, but rather on enabling an untrusted prover to provide a guarantee to a verifier that the prover performed a requested computation correctly.

References

[ALM$^+$98] 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.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *Journal of the ACM (JACM)*, 45(1):70–122, 1998.

[Bab85] László Babai. Trading group theory for randomness. In Robert Sedgewick, editor, *STOC*, pages 421–429. ACM, 1985.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

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

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

[Sha90] Adi Shamir. Ip=pspace. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 11–15. IEEE Computer Society, 1990.