The Bright Side of Hardness:

Relating Computational Complexity and Cryptography

(notes for an overview talk)

Oded Goldreich
Department of Computer Science
Weizmann Institute of Science, Israel.
oded.goldreich@weizmann.ac.il

July 12, 2008

This document contains preparation notes for a 45-minute talk on the relation between computational difficulty and cryptography. References and further details can be found in the author's texts (e.g., a primer [4] and two-volume book [3]).

Contents

1 (title page)

This is my 1st ever PowerPoint presentation. I hope I'll manage it.

2 Background: The P versus NP Question

I want to start from the basics, which is the fundamental *P versus NP Question*.

Note that $\mathcal{P} \neq \mathcal{NP}$ means that $\langle$`first two lines on the slide`$\rangle$.

Philosophically, this guarantees the meaningfulness of the notion of a problem and of the notion of a proof.

Real-life implications: the bad news (dark side), and the good news (bright side).

3 One-Way Functions

The good/bad news depend on *typical* (i.e., average-case) hardness, since rare cases of hardness are irrelevant to real-life.

Actually, for the good news we need slightly more: we need the ability to generate hard instances coupled with solutions (known to us but not to others). This leads to the definition of one-way functions (where the problem instance is the image, and a solution is a preimage).

On the notion of *negligible*: robust under a feasible number of repeations. (Actually, defined so.)

Mention: the `MULTiplication` example.

4 Applications to Cryptography

(This is a kind of preview to the rest of the talk.)

- The classical tasks $\longrightarrow$
- A major modern task $\longrightarrow$
-
- A major modern task $\longrightarrow$
- A general result (of amazingly wide scope) $\longrightarrow$

5 Amplifying Hardness

In order to obtain these applications, we need to amplify hardness.

Note that, by def., a OWF does NOT necessarily *hide* any part of the preimage in a *perfect manner*, but rather frustrates the *reconstruction of the entire preimage* (via feasible methods). In contrast, a hardcore is a "part of the preimage" that is perfectly hidden (w.r.t feasible methods).

N.B.: Any predicate can be easily guesses with success probability 1/2. A hardcore predicate cannot be (feasiblly) guessed significantly better.

N.B.: In the special case of 1-1 function f, having a hardcore implies that f is hard to invert (otherwise guessing is feasble via inverting f and evaluating b).

6 The Existence of a Hardcore

Note: f' is a slight modification of f.

Re the warm-up: We'll just prove that b is hard to guess w.p. at least 0.76 (rather than at least 0.51). Assuming towards the contradiction that b can be guessed with error probability at most 0.24, we show how to guess each bit of the preimage with error probability at most 0.48 (and we can get negligible error via repetitions and ruling by majority, which allows inverting f').

7 1st Application: Pseudoerandom Generators

The application of hardness is hinted in the THM.

As we shall see in a few slides, pseudoerandom generators are central to Cryptography. But they are also of independent interest (i.e., beyond Crypto.).

The elements of this notion are: (1) efficient evaluation, (2) stretch, and (3) computational indistinguishability from true randomness.

(formal definition – in next slide)

8 Pseudoerandom Generators (form.)

Highlight: Any (reasonable) stretch is equivalent (w.r.t existence of such PRGs) to minimal stretch. That is, all (reasonable) stretches are equivalent (w.r.t existence of such PRGs).

Highlight/THM: there exist PRGs iff there exist OWFs. (see next slide)

9 PRG iff OWF ("Hardness vs Randomness")

Highlight: the "Hardness vs (Pseudo)Randomness" trading. By definition, pseudorandomness refers to distributions that are statistically far apart but are *infeasible* to tell apart. Furthermore, the THM (via its effective proof) shows how hardness (i.e., OWF) can be converted to the generation of (pseudo)randomness, and vice versa.

Explain (PRG implies OWF): Inverting f is feasible on G's output (by contra. hypo.), but inverting f is not possible (except w. negl. prob.) on random strings (because these are not in f's image).

Explain (OW**P**[1] implies PRG): For each bit position, the next bit in G's output is hard to predict (by triviality if $i < |s|$ and by hardcode if $i = |s|$).

Explain (re stretch): maximal stretch means the maximal output length that allows merely printing the output within feasible time. That is, printing $|G(s)|$-many ones should be feasible (in order to allow meaningful talk of effecting $s \mapsto G(s)$).

[1] OWF that is 1-1 (and length preserving).

10 Pseudoerandom Functions

By PRF we mean a collection of functions, sharing an evaluation algorithm (as in item 1) and failing "Turing's Test of Randomness" (as in item 2).

Historical comments (see also extra slide): Item 2 is called "Turing's Test of Randomness" in analogy to "Turing's Test of Intelligence": distinguishing a computer program from a Human via an Q&A interaction. Yet, even more related to Turing is the following quote from Turing's work (1950): *I have set up on a Manchester computer a small programme using only 1000 units of storage, whereby the machine supplied with one sixteen figure number replies with another within two seconds. I would defy anyone to learn from these replies sufficient about the programme to be able to predict any replies to untried values.*

THM: Can build PRF based on any PRG.

11 Cryptography: private-key encryption (based on PRF)

The problem: Alice and Bob wish to communicate in privacy over a communication channel that may be tapped by an adversary. They must know something that the adversary does not know, which is called a key. Bob transforms the messages he want to send to Alice by using an encryption algorithm (using their shared key), resulting in ciphertexts sent over the channel. Alice reconstructs the messages by using a decryption algorithm (with the same key), but the adversary reading the ciphertexts should learn nothing about the original messages.

The solution: encrypt the message by XORing it with the value of the PRF evaluated at a random point, which is also placed in the ciphertext. In fact, it suffices to use a different evaluation point in each encryption.

12 Cryptography: message authentication (based on PRF)

The problem: Forget about privacy, here Alice wants to be sure that the message she receives was actually sent by Bob (rather than injected on the channel by an adversary). Again, Alice and Bob must share a key, which the adversary does not know. Bob appends to his messages a suitable "authentication tag" which can be verified by Alice, where the parties use suitable signing and verifying algorithms (and their shared key). The adversary reading prior message–tag pairs should not be able to generate a new valid message–tag pair (i.e., a new pair that passes verification).

The solution: generate and verify tags by applying a PRF to the message.

13 More Cryptography: Sign and Commit

Signature schemes are like message authentication except that they allow *universal* verification (by parties not holding the signing key, but rather having access to public verification keys).

Commitment schemes (to be used in the sequel...) are two-party protocols consisting of two phases such that after the first phase ("commit phase") the sender is "committed" to a value (which can be revealed later) but the receiver remains oblivious of that value. (These schemes satisfy conflicting hiding and binding properties.)

THM: OWF (equiv., PRG) imply both Signature and Commitment schemes.

14 A generic cryptographic task: forcing parties to follow presecribed instructions

Note: forcing parties to follow presecribed instructions makes sense "only" in cryptographic settings, and it presumes that such actions depend on private information (or else others can effect them and verification is trivial...).

Also note that it makes no sense to ask whether the party has used its "true" input (which is in its head); what makes sense is (e.g.,) requiring that the message sent is consistent with some private input.

Clearly, the above can be proved by revealing the private input, but this eliminates the entire point of cryptography (of protecting the privacy of party's inputs).

What we wish is to prove that such an input x exists *without revealing anything else about it*. The "idea" is that such proof systems, called *zero-knowledge*, can be (formally defined and) constructed.

THM: Commitment schemes imply zero-knowledge proofs for every such (erfficient) predetermined computation.

15 Zero-Knowledge Proof Systems

E.g., for graph 3-colorability (which is NP-complete).

Task: prove that a graph is 3-colorable without revealing anything else (beyond what follows easily from this fact).

The protocol uses commitments to possible colors (i.e., ternary values) of the various vertices. Argue about (1) completeness, (2) soundness, and (3) zero-knowledge.

THM: Commitment schemes imply zero-knowledge proofs for 3-colorability (and any other problem in NP).

16 Universal Results: general secure multi-party computations

Outrageous claim: any desired multi-party functionality can be implemented securly. This claim represents a vaiety of THMs that relate to various models, some rely on computational assumptions (e.g., OWF or rather TDP).

What do I mean by a multi-party functionality? Any (efficient) probabilistic process that maps m-ary sequences of local inputs to m-ary sequences of local outputs is such a (desired) functionality. Restricted to deterministic functionalities, this means a sequence of functions $f_1, ..., f_m$ that determine the desired local output of each party as a function of all local inputs (i.e., the i^{th} party should obtain the output $f_i(x)$, where $x = (x_1, ..., x_m)$ and x_j denotes the input of Party j).

In other words, we can obtain the same effect as when each party sends its private input to an imaginary *trusted party*, which computes and properly distrubted the corresponding outputs. Indeed, *the effect of an imaginary trusted party can be securely emulated by the mutually distrustful parties.*

References

[1] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SICOMP*, Vol. 13, pages 850–864, 1984. Preliminary version in *23rd FOCS*, 1982.

[2] W. Diffie, and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22 (Nov. 1976), pages 644–654.

[3] O. Goldreich. *Foundation of Cryptography*, in two volumes: *Basic Tools* and *Basic Applications*. Cambridge University Press, 2001 and 2004.

[4] O. Goldreich. Foundations of Cryptography – A Primer. *Foundations and Trends in Theoretical Computer Science*, Volume 1, Issue 1, 2005.

[5] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[6] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *Journal of the ACM*, Vol. 33, No. 4, pages 792–807, 1986.

[7] O. Goldreich and L.A. Levin. Hard-core Predicates for any One-Way Function. In *21st STOC*, pages 25–32, 1989.

[8] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, Vol. 38, No. 3, pages 691–729, 1991. Preliminary version in *27th FOCS*, 1986.

[9] O. Goldreich, S. Micali and A. Wigderson. How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In *19th ACM Symposium on the Theory of Computing*, pages 218–229, 1987.

[10] S. Goldwasser and S. Micali. Probabilistic Encryption. *JCSS*, Vol. 28, No. 2, pages 270–299, 1984. Preliminary version in *14th STOC*, 1982.

[11] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, Vol. 18, pages 186–208, 1989. Preliminary version in *17th ACM Symposium on the Theory of Computing*, 1985. Earlier versions date to 1982.

[12] S. Goldwasser, S. Micali, and R.L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, April 1988, pages 281–308.

[13] J. Håstad, R. Impagliazzo, L.A. Levin and M. Luby. A Pseudorandom Generator from any One-way Function. *SICOMP*, Vol. 28, No. 4, pages 1364–1396, 1999. Preliminary versions by Impagliazzo *et. al.* in *21st STOC* (1989) and Håstad in *22nd STOC* (1990).

[14] M. Naor. Bit Commitment using Pseudorandom Generators. *Journal of Cryptology*, Vol. 4, pages 151–158, 1991.

[15] R. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *CACM*, Vol. 21, Feb. 1978, pages 120–126.

[16] J. Rompel. One-way Functions are Necessary and Sufficient for Secure Signatures. In *22nd ACM Symposium on the Theory of Computing*, 1990, pages 387–394.

[17] A.C. Yao. Theory and Application of Trapdoor Functions. In *23rd FOCS*, pages 80–91, 1982.