Sorting out zero-knowledge

Gilles BRASSARD†
Département IRO
Université de Montréal

Claude CREPEAU‡
Laboratory for Computer Science
Massachusetts Institute of Technology

1. Introduction

In their 1985 paper, Goldwasser, Micali and Rackoff set forth the notion of zero-knowledge interactive proofs [GMR1]. This seminal paper generated considerable activity around the world. In the span of a few years, a substantial number of results were obtained by different groups of researchers. Among those, the following two theorems make an intriguing pair:

- *Fortnow* [F], together with *Boppana, Hastad and Zachos* [BHZ]: The existence of a perfect zero-knowledge protocol for an **NP**-complete problem would imply that the polynomial hierarchy collapses.
- *Brassard, Chaum and Crépeau* [BCC]: There exists a perfect zero-knowledge protocol for satisfiability.

Nevertheless, the polynomial hierarchy has *not* collapsed! Of course, the resolution of this apparent paradox is that the above two results strongly depend on fundamentally incompatible definitions of what a protocol is.

The purpose of this paper is to explain the various notions involved and to offer a new terminology that emphasizes their differences. There are two *orthogonal* aspects to zero-knowledge interactive proofs. One is the notion of zero-knowledge and the other is the notion of interactive proof. Unfortunately, these two notions are often thought to be inseparable. This confusion is reminiscent of the long lasting confusion among many people between public-key encryption and digital signature. It is clear that interactive proofs make sense independently of zero-knowledge (after all, Babai's Arthur–Merlin games [Ba] were invented independently of [GMR1]), but it is more subtle to see that a protocol could be zero-knowledge without being an interactive

† Supported in part by Canada NSERC grant A4107.

‡ Supported in part by an NSERC postgraduate scholarship; part of this research was performed while this author was visiting the Université de Montréal.

proof.

2. Proofs and Arguments

According to [GMR1], an interactive proof is a protocol that takes place between an arbitrarily powerful prover and a computationally limited verifier. The purpose of such protocols is for the prover to convince the verifier of the validity of an assertion. This notion is very similar to Babai's Arthur–Merlin games [Ba, BM] (not discussed here). As essential aspect of these protocols is that there is nothing a prover can do in order to convince the verifier of a false statement, except to blindly hope for an exponentially small probability of lucky successful cheating. This is the sense in which it is said of GMR interactive proofs that ''A proof is a proof''.

In many practical situations, it is nevertheless reasonable to impose computational limitations on the prover. This setting was investigated independently by Chaum [C2] (who, on the other hand, allowed unlimited computing power to the verifier — to a large extent, Chaum's model goes back to [C1]), and by Brassard and Crépeau [BC] (who restricted both the prover and the verifier to ''reasonable'' computing power). In either the Chaum or the Brassard–Crépeau setting, an interactive ''proof'' may not be a proof at all. Indeed, a computationally unbounded prover could cheat in convincing the verifier of a false statement. More importantly in practice, many of these ''proofs'' rely on an unproved assumption, usually the computational difficulty of computing some specific problem. For instance, even a polynomial-time prover could succeed at ''proving'' a false statement with the protocol of [BC], provided that she has a very efficient factoring algorithm (whose non-existence is still an open problem).

To distinguish interactive proofs in the sense of GMR from the protocols of [C2, BC], we introduce a new terminology. An interactive protocol is an *argument* (rather than a proof) if the verifier's faith in the prover's claim must ultimately rest on an assumption. As we have seen, this assumption could be cryptographic in nature, such as the assumption that the prover cannot compute a discrete logarithm while the protocol is in progress [BCC]. It could also be merely that the prover has time-bounded resources (such as polynomial time), although our current state of knowledge in computational complexity does not yet allow us to make use of this kind of assumption alone. The assumption behind an argument could also be of a mathematical or even physical nature, such as the Extended Riemann Hypothesis or the principles of quantum physics [BB]. Or it could be that there are two provers who initially agree on their strategy, but with the assumption that they cannot communicate while the protocol is in progress [BGKW, GKBW]. Other types of assumptions can be dreamt up as well.

The first published use of this new terminology appeared in the title of [BCY]. It should be pointed out that what we now call argument has occasionally been termed *pseudo-proof* by others. We wish to make a statement here, as the inventors of the notion (co-invented by Chaum), that we **very strongly** object to the appellation of pseudo-proof for the protocols of [C2, BC, BCY].

One may well wonder why settle for arguments when interactive proofs are so much stronger. One reason has to do with the possible zero-knowledge aspect of these protocols (see next section): it is unlikely that perfect zero-knowledge interactive proofs exist for statements concerning **NP**-complete problems [F, BHZ], and even computational zero-knowledge interactive proofs for such statements are not known to exist unless one makes cryptographic assumptions [GMW]. In contrast, perfect zero-knowledge arguments are known for all such statements, and no assumptions are needed to prove that these arguments are perfect zero-knowledge [BCC] — but of course they are not interactive proofs at all in the sense of GMR.

To a large extent, the (probably) computational zero-knowledge interactive proofs of [GMW] and the perfect zero-knowledge arguments of [BCC] are duals of each other. The only aspect of these protocols for which duality fails is that a cheating verifier can work off-line on the transcript of a [GMW] interactive proof and spend as much time as he wishes in attempts to decipher the prover's secret. In contrast, the prover can only cheat a [BCC] argument if she can break the cryptographic assumption on-line, while the protocol is in progress. As a consequence, an algorithm capable of breaking the cryptographic assumption in a few months of CRAY computing time would be bad news for [GMW] proofs but of no real consequence for [BCC] arguments. The reader is referred to section 7 of [BCC] for a more complete discussion of this issue.

Another reason to restrict the computational power of the prover is that it allows one to make sense of the notion of ''proof of knowledge'' [FFS, TW]. For instance, one such protocol allows a prover to convince a verifier that she (the prover) knows the factors of a given large integer [T]. The point is that it would be absurd for a prover known for her unbounded computing power to convince a verifier that she knows those factors: *of course* she knows them since she can compute them whenever she wishes. These proofs of knowledge play a crucial role in modern identification schemes [FFS] (but read [BBDGQ]). (You may wonder at this point why we call them ''proofs of knowledge'' rather than ''arguments of knowledge'' even though the prover's computing power is limited. For one thing, this terminology is well established. More importantly, they *are* GMR-proofs since no assumptions are needed for the proofs to be valid; it is only that they would be of no interest whatsoever if performed by a prover with unbounded computing power or if factoring, in our example, were known to be easy.)

For more discussion on the difference between the settings of [GMR1,2], [C2] and [BC], the reader is referred to [BCC, p. 159].

3. Zero-knowledge

The notion of zero-knowledge was set forward by Goldwasser, Micali and Rackoff [GMR1]. The reader is encouraged to read also the subsequent journal version of their work [GMR2]. Essentially, a zero-knowledge protocol allows a prover to convince a verifier of an assertion without disclosing any information to the verifier beyond the validity of that assertion. In the context of [GMR1,2], all zero-knowledge protocols are interactive proofs, but the notion of zero-knowledge *arguments* makes perfect sense as well. Whenever the prover's computing power is limited, a zero-knowledge protocol will necessarily disclose more than the validity of the assertion: the fact that the prover knows why this assertion is valid is also disclosed. (In the context of interactive proofs, the fact that the prover has this knowledge is implied by her unbounded computing power.) Nevertheless, this additional piece of information revealed when the prover's computing power is limited makes it possible to design protocols that actually reveal *less* than would be possible for any (interesting) interactive proof in which the prover has unbounded computing power: these are the proofs of knowledge discussed at the end of the previous section.

The intent of this paper is not to repeat the formal definitions of zero-knowledge, which can be found in [GMR2]. Rather, we wish to describe briefly several subtly different definitions for the main purpose of contrasting them. The reader not already familiar with the notion of (zero-knowledge) *simulator* is urged to read [GMR2] in order to profit fully from what follows, even though we briefly recall this notion in the next two paragraphs (with apologies to those who *are* familiar with it).

A protocol is perfect zero-knowledge [GMW] if the verifier does not learn anything at all from the interaction beyond the validity of the assertion involved and —if relevant— the fact that the prover knows why it is valid. In order to define this notion more formally, one has to consider the *view* of what the verifier sees during his interaction with the prover. This consists of the outcome of his own random coin tosses as well as of everything that the prover tells him during the interaction. Because of the probabilistic nature of interactive protocols (including random choices made by the prover), a probability distribution is defined on the view of the verifier. A protocol is *perfect* zero-knowledge if, to each polynomial-time verifier, there corresponds a polynomial-time *simulator* capable of producing a view taken from exactly the same probability distribution *without ever talking to the prover*. Intuitively, the existence of this simulator shows that the verifier does not learn anything from the interaction since the prover does not tell him anything that he could not have produced by himself (probabilistically speaking).

If the simulator is only required to produce a view taken from a probability distribution that is polynomially indistinguishable [GM] from the ''correct distribution'', the protocol is said to be *computational* zero-knowledge [GMR1]. Intuitively, this means

that whatever the verifier may obtain from the interaction, he cannot make use of in polynomial time. Finally, the protocol is *statistical* (or *almost perfect*) zero-knowledge [F] if the real and simulated probability distributions are statistically close. When a protocol is simply said to be ‘‘zero-knowledge’’, most authors mean that it is computational zero-knowledge. In the opinion of the current authors, it would be better if the default option were that such protocols are perfect zero-knowledge.

Some researchers have objected to calling *zero* knowledge a protocol that reveals one bit of knowledge, for instance that a given graph is Hamiltonian. For this reason, Galil, Haber and Yung have suggested that such protocols be termed *minimum* knowledge [GHY], meaning that they reveal nothing more than what they absolutely have to by definition of their purpose. By contrast, Feige, Fiat and Shamir wished to keep the term ‘‘zero-knowledge’’, so they introduced the notion of proof of knowledge in order to have a protocol that is truly *zero* knowledge (in their opinion) [FFS] in the sense that it reveals zero bits of information about the ‘‘real world’’. This notion of proof of knowledge was also formalized by Tompa and Woll [TW].

The original definition of zero-knowledge [GMR1] lacked the desirable property that the sequential composition of two zero-knowledge protocols should remain zero-knowledge [GK]. If this property is important, the stronger notion of zero-knowledge *with auxiliary input* [O] should be used. A protocol is perfect zero-knowledge with auxiliary input if, no matter which additional input is given to the verifier before the interaction with the prover, the interaction does not help the verifier learn anything at all that he could not have learnt by himself given the same additional input. The notions of computational and statistical zero-knowledge with auxiliary input are defined similarly.

The formal definition of zero-knowledge [GMR1,2] stipulated that a polynomial-time simulator should exist for each polynomial-time verifier. This begs the following two questions.

1) Should there be a uniform and efficient process by which the simulator can be derived from the verifier?

2) How should the definition be modified if one is also interested in dealing with verifiers that are not limited to polynomial time? (This is an important issue when dealing with arguments rather than interactive proofs.)

We believe that the answer to the first question ought to be ‘‘yes’’. Otherwise, it is hard to support the view that a verifier learns nothing from the interaction if his conversation with the prover could be simulated but only by a simulator that is infeasible to find. We can think of two different definitions that would force the simulator to be efficiently obtainable from the verifier. The first and most obvious definition requires that the verifier be provided using a fixed formalism, such as that of probabilistic interactive Turing machines. In this case, we would insist that the ‘‘code’’ for

the simulator (presumably expressed in the same formalism) be efficiently derivable from the code for the verifier.

The second definition, which we prefer for its simplicity even though it is more restrictive, is that of a ‘‘black box simulator’’, first introduced by Oren [O]. This definition requires the simulator to use the verifier as a resettable black box in order to simulate the interaction between that verifier and the prover. In other words, the simulator has no access to the code of the verifier, but has control over its tapes, including its random coin flip tape, and has the ability to bring it to a halt and restart it in its starting state (possibly with different tapes) at any time it wishes. This definition is not as restrictive as it might appear because most simulators in the literature are in fact designed in the black box model. It is known that black box simulation implies zero-knowledge with auxiliary input [O].

This brings us to the second question mentioned above: ‘‘How should the definition be modified if one is also interested in dealing with verifiers that are not limited to polynomial time?’’. One possibility would be to grant the simulator an amount of time comparable (perhaps up to a polynomial factor) to the time allowed to the verifier. Another possibility, which we prefer, involves again the notion of black box simulator: restrict the simulator to polynomial time, but count at unit cost any call on the verifier.

Thus we see that there are two different reasons why there may be protocols that are GMR zero-knowledge but not black box zero-knowledge. Firstly, there may be protocols that are non-uniform zero-knowledge but that cannot be simulated by a black box simulator. Moreover, given the work of [GK], it is not hard to design a protocol that is zero-knowledge against any polynomial-time verifier (although not with auxiliary input), but that would nevertheless allow a super-polynomial-time verifier to get information from an even more powerful prover that he could not compute by himself within his allowed time bound.

All the concepts discussed so far are but minor variations on the original definition of zero-knowledge [GMR1]. Other minor variations have been introduced, such as the notion of zero-knowledge *with respect to a trusted verifier*. We do not discuss them here. Rather, we now discuss a few notions that are more significantly different.

There are practical situations in which it is worthwhile to settle for something *less* than zero-knowledge. This may be the case, for instance, if it allows a substantial increase in efficiency. A nice example of this concept is the version of their identification scheme that Feige, Fiat and Shamir discuss in section 4 of [FFS]. They propose a scheme that is (probably) not zero-knowledge because the interaction reveals information to the verifier that he could (probably) not have computed by himself. Nevertheless, this information is proven in [FFS] not to be enough to help the verifier cheat the system in polynomial time, provided that cheating the protocol in polynomial time is impossible in the first place without the interaction. In other words, partial

information is perhaps released on the prover's secret, but this partial information is not sufficient for the verifier to recompute the prover's secret (or any ''equivalent'' secret that might allow him to defeat the system). Feige, Fiat and Shamir say of such schemes that they ''release no useful information''.

A similar but stronger notion was introduced by Brassard, Chaum and Crépeau [BCC]. A protocol is *minimum disclosure* if everything that the prover ever says to the verifier is uncorrelated to her secret (or any equivalent secret). Clearly, a minimum disclosure protocol cannot help the verifier compute the prover's secret (or any equivalent secret), hence it ''releases no useful information''. However, it may not be zero-knowledge because the prover is allowed to reveal irrelevant information even if this information would be hard for the verifier to compute by himself. Contrary to protocols that reveal no useful information, a minimum disclosure protocol is not allowed to reveal partial information, even if such information is not enough to efficiently recompute the secret.

This brings us to a curious notion, which has evolved from discussions between the authors and Silvio Micali. Some protocols are not known to be zero-knowledge nor even minimum disclosure. However, it *is* known that whatever the verifier could learn by deviating arbitrarily from his prescribed behaviour could not be worth more, for instance, than the discrete logarithm of a number of his choosing. (This is the case, in particular, of the parallel version of the main protocol given in [BCC] if the blobs are implemented as suggested in section 6.1.2.) This is *not* to say that the verifier can actually obtain this discrete logarithm from his interaction with the prover, but whatever he can get is worth no more than if he did. Formally, this is so because it would be easy for a simulator to do a perfect job very efficiently were it only given this discrete logarithm. If it can be proven that this discrete logarithm is uncorrelated to the prover's secret (or any equivalent secret), then the corresponding protocol is minimum disclosure. We would like to point out that, because such correlation cannot be ruled out in general, the claim made in [BCC, p. 166] to the effect that ''the parallel version of the protocol is minimum disclosure'' is not known to hold in all cases.

Given a protocol such as the one we have just discussed, the interesting question is to determine the minimal piece of information that would allow efficient simulation. This information can be thought of as the maximal value of the protocol for any cheating verifier because whatever a verifier can get from the interaction is worth no more than it. An interesting question asked by Silvio Micali is what is the maximal value of the parallel version of the protocol for graph isomorphism [GMW]? Could it be less than the isomorphism itself?

4. Complexity issues

There are several complexity issues pertaining to zero-knowledge protocols. One of the most interesting from a practical point of view is the issue of parallelism. Zero-knowledge protocols usually take place in several rounds of interaction between the prover and the verifier. Each round increases the verifier's confidence in the prover's honesty. Most of these protocols can be reformulated in a natural way in a bounded number of rounds, but they apparently cease to be zero-knowledge when this is done. (As previously mentioned, the identification protocol of [FFS] is an interesting special case: its parallel version may not be zero-knowledge but it releases no useful information, which is just as good for the application that they have in mind.) Nevertheless, it is known that the general protocols of [GMW] and [BCC] *can* be redesigned to be run in parallel and remain zero-knowledge (perfect zero-knowledge in the case of [BCC]). For a full discussion of the fact that everything in **NP** can be argued in perfect zero-knowledge in a bounded number of rounds, read the ICALP version of [BCY]. A summary, including the history of the problem and references to related work, can be found in these EUROCRYPT '89 Proceedings. Following the work of [BCY], Bellare, Micali and Ostrovsky have shown than graph isomorphism can be *proven* in perfect zero-knowledge in a bounded number of rounds (personal communication).

An orthogonal result pertaining to bounded round interactive proofs is due to Fortnow [F] and to Aiello and Hastad [AH]: if L is any language that admits a perfect or statistical zero-knowledge interactive proof system with an arbitrary number of rounds, then both L and its complement also admit a *bounded* round interactive proof system (which is not zero-knowledge in general, however). It is crucial for the Fortnow and Aiello–Hastad results not to limit the prover to polynomial time.

Despite the work mentioned in the first paragraph of this section, it remains interesting to analyse what happens if one runs an arbitrary zero-knowledge protocol in parallel without any additional precaution. In particular, it has not been proven that these protocols cease to be zero-knowledge (it is only the case that no one has been able to design a simulator capable of coping with the situation). If indeed these protocols are not zero-knowledge, what is in general their maximal value (as defined at the end of the previous section)? When are they minimum disclosure? When do they release no useful information?

Other complexity issues pertain to the simulator. One of them is whether it is legitimate to say that a protocol is zero-knowledge if there exists a verifier capable in quadratic time to force the prover to take part in an interaction that no simulator could reproduce in less than cubic time.

Yet another issue pertaining to the simulator is that of *strict* polynomial time versus *expected* polynomial time. Recall that both the verifier and the simulator are probabilistic processes. It is usually the case that the prescribed verifier takes a time

that is strictly bounded by a fixed polynomial, regardless of its random choices (perhaps the only exception in the published literature is the prescribed verifier for the zero-knowledge interactive protocol for graph 3-colourability when the number of edges is not a power of 2 [GMW]). Nevertheless, the simulator usually requires polynomial time in the expected sense. Is it possible in general to make do with strict polynomial time for the simulator whenever the verifier is strict polynomial time (or when the black box model is used)? In particular, it is an open question to design a strict polynomial-time simulator that retains the *perfect* zero-knowledge property of the well-known protocol for graph isomorphism [GMW].

ACKNOWLEDGEMENTS

The idea of writing this paper came from several discussions the authors have had with Silvio Micali.

BIBLIOGRAPHY

[AH] Aiello, W. and Hastad, J., ‘‘Perfect zero-knowledge languages can be recognized in two rounds’’, *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, 1987, pp. 439–448.

[Ba] Babai, L., ‘‘Trading group theory for randomness’’, *Proceedings of the 17th ACM Symposium on Theory of Computing*, 1985, pp. 421–429.

[BM] Babai, L. and Moran, S., ‘‘Arthur–Merlin games: A randomized proof system, and a hierarchy of complexity classes’’, *Journal of Computer and System Sciences*, vol. 36, 1988, pp. 254–276.

[BBDGQ] Bengio, S., Brassard, G., Desmedt, Y., Goutier, C. and Quisquater, J.–J., ‘‘Secure implementation of identification systems’’, in preparation.

[BB] Bennett, C. H. and Brassard, G., ‘‘Quantum cryptography’’, in preparation; in the mean time, read chapter 6 in [Br].

[BGKW] Ben Or, M., Goldwasser, S., Kilian, J. and Wigderson, A., ‘‘Multi-prover interactive proofs: How to remove intractability assumptions’’, *Proceedings of the 20th ACM Symposium on Theory of Computing*, 1988, pp. 113–131.

[BHZ] Boppana, R. B., Hastad, J. and Zachos, S., ‘‘Does co–**NP** have short interactive proofs?’’, *Information Processing Letters*, vol. 25, 1987, pp. 127–132.

[Br] Brassard, G., *Modern Cryptology: A Tutorial*, Lecture Notes in Computer Science, vol. 325, Springer-Verlag, 1988.

[BCC] Brassard, G., Chaum, D. and Crépeau, C., ‘‘Minimum disclosure proofs of knowledge’’, *Journal of Computer and System Sciences*, vol. 37, no. 2, 1988, pp. 156–189.

[BC] Brassard, G. and Crépeau, C., ‘‘Non-transitive transfer of confidence: A *perfect* zero-knowledge interactive protocol for SAT and beyond’’, *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, 1986, pp. 188–195.

[BCY] Brassard, G., Crépeau, C. and Yung, M., ‘‘Everything in **NP** can be argued in *perfect* zero-knowledge in a *bounded* number of rounds’’, *Proceedings of 16th ICALP Conference*, Stresa, Italy, July 1989, to appear; an extended abstract appears in these *EUROCRYPT ’89 Proceedings*.

[C1] Chaum, D., ‘‘Security without identification: Transaction system to make Big Brother obsolete’’, *Communications of the ACM*, vol. 28, 1985, pp. 1030–1044.

[C2] Chaum, D., ‘‘Demonstrating that a public predicate can be satisfied without revealing any information about how’’, *Advances in Cryptology – CRYPTO ’86 Proceedings*, Springer-Verlag, 1987, pp. 195–199.

[FFS] Feige, U., Fiat, A. and Shamir, A., ‘‘Zero knowledge proofs of identity’’, *Journal of Cryptology*, vol. 1, no. 2, 1988, pp. 77–94.

[F] Fortnow, L., ‘‘The complexity of perfect zero-knowledge’’, *Proceedings of the 19th ACM Symposium on Theory of Computing*, 1987, pp. 204–209.

[GHY] Galil, Z., Haber, S. and Yung, M., ‘‘A private interactive test of a Boolean predicate and minimum-knowledge public-key cryptosystems’’, *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, 1985, pp. 360–371.

[GK] Goldreich, O. and Krawczyk, H., ‘‘On sparse pseudo-random distributions’’, *Advances in Cryptology – CRYPTO ’89 Proceedings*, Springer-Verlag, to appear.

[GMW] Goldreich, O., Micali, S. and Wigderson, A., ‘‘Proofs that yield nothing but their validity and a methodology of cryptographic protocol design’’, *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, 1986, pp. 174–187.

[GKBW] Goldwasser, S., Kilian, J., Ben Or, M. and Wigderson, A., ‘‘Efficient identification schemes using two prover interactive proofs’’, *Advances in Cryptology – CRYPTO ’89 Proceedings*, Springer-Verlag, to appear.

[GM] Goldwasser, S. and Micali, S., ‘‘Probabilistic encryption’’, *Journal of Computer and System Sciences*, vol. 28, 1984, pp. 270–299.

[GMR1] Goldwasser, S., Micali, S. and Rackoff, C., ‘‘The knowledge complexity of interactive proof systems’’, *Proceedings of the 17th ACM Symposium on Theory of Computing*, 1985, pp. 291–304;

[GMR2] Goldwasser, S., Micali, S. and Rackoff, C., ‘‘The knowledge complexity of interactive proof systems’’, *SIAM Journal on Computing*, vol. 18, no. 1, 1989, pp. 186–208.

[O] Oren, Y., ‘‘On the cunning power of cheating verifiers: Some observations about zero knowledge proofs’’, *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, 1987, pp. 462–471.

[T] Tompa, M., ‘‘Zero knowledge interactive proofs of knowledge (a digest)’’, *Second Conference on Theoretical Aspects of Reasoning about Knowledge*, Monterey, CA, 1988; available as Research Report RC 13282 (#59389), IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY, 1987.

[TW] Tompa, M. and Woll, H., ‘‘Random self-reducibility and zero-knowledge interactive proofs of possession of information’’, *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, 1987, pp. 472–482.