JOURNAL OF COMPUTER AND SYSTEM SCIENCES 37, 156–189 (1988)

Minimum Disclosure Proofs of Knowledge

GILLES BRASSARD*

Département d'informatique et de R.O., Université de Montréal, C.P. 6128, Succursale "A," Montréal, Québec, Canada H3C 3J7

DAVID CHAUM

Centre for Mathematics and Computer Science (CWI), Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

AND

CLAUDE CRÉPEAU[†]

Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, Massachusetts 02139

Received July 3, 1987

Protocols are given for allowing a "prover" to convince a "verifier" that the prover knows some verifiable secret information, without allowing the verifier to learn anything about the secret. The secret can be probabilistically or deterministically verifiable, and only one of the prover or the verifier need have constrained resources. This paper unifies and extends models and techniques previously put forward by the authors, and compares some independent related work. © 1988 Academic Press, Inc.

1. INTRODUCTION

Assume Peggy ("the prover") knows some information. For instance, this could be the proof of a theorem or the prime factorization of a large integer. Assume further that Peggy's information is *verifiable*, in the sense that there exists an efficient procedure capable of certifying its validity. In order to convince Vic ("the verifier") of this fact, Peggy could simply reveal the information to him so that he could perform the certifying procedure himself. This would be a *maximum disclosure* proof, since it results in Vic learning all the information. He could therefore later show it to someone else and even claim it to have been his originally.

* Supported in part by NSERC Grant A4107. Research conducted in part at CWI.

[†] Supported in part by an NSERC postgraduate scholarship. Supported in part by NSF Grant DCR MCS8509905. Research conducted in part at Université de Montréal and at CWI.

0022-0000/88 $3.00

In this paper we give a general protocol for obtaining *minimum disclosure* proofs and several practical ways to implement it. This protocol allows Peggy to convince Vic, beyond any reasonable doubt, that she has information that would pass the certifying procedure, but in a way that does not help him determine this information. For example, if Peggy's information is the proof of a theorem, Vic is left with the conviction that Peggy knows how to prove it, and hence that the theorem is true. However, Vic is not given even a clue as to how the proof might proceed (except perhaps for an upper limit on its length). Although Peggy's original information is verifiable, the conviction thus obtained by Vic may not be. In particular, conducting the protocol with Peggy need not (and in many cases *will* not) enable Vic to subsequently convince someone else.

The notion of minimum disclosure proofs extends to the case of *probabilistically verifiable information.* Assume, for instance, that Peggy generates two integers that are almost certainly prime according to some probabilistic algorithm [R1, SS]. She computes their product, makes it public, and then claims that she knows its prime factorization. Does she have verifiable information to support her claim, considering the fact that she does not have a definite proof that her factors are prime? In a case like this, even though the efficient certifying procedure for her information is probabilistic, it still makes sense for her to use a minimum disclosure proof to convince Vic that her claim is true. Our minimum disclosure proof techniques extend to the case of probabilistically verifiable information.

At the heart of all our protocols is the notion of *bit commitment*, which allows Peggy to commit herself to the value of some bits in a way that prevents Vic from learning them without her help. Bit commitment is implemented through our main primitive, which we call for convenience the "*blob.*" As this paper shows, the blob is a *universal* primitive for minimum disclosure. Each blob is used by Peggy as a commitment to either 0 or 1. For the sake of generality, we do not impose any restriction on the nature of blobs—they could be made out of fairy dust if this were useful. By "Peggy commits to a blob," we mean that Peggy has a blob "in mind" and that she does something that will force her to stick to this blob in the future. If the blob itself can be represented as a bit string—as in most practical cases—committing to a blob can be as simple as showing it in the clear. The abstract defining properties of blobs are as follows:

(i) Peggy can commit to blobs: by committing to a blob, she is in effect committing to a bit.

(ii) Peggy can *open* any blob she has committed to: she can convince Vic of the value of the bit she in effect committed to when she committed to the blob. Thus, there is no blob she is able to "open" both as 0 and as 1.

(iii) Vic cannot learn anything about which way Peggy is able to open any unopened blob she has committed to. This remains true even after other blobs have been opened by Peggy.

(iv) Blobs do not carry "side information": the blobs themselves as well as the processes by which Peggy commits to and opens them are uncorrelated to any secret she wishes to keep from Vic.

Consider the following illustrative implementation of a blob. When Peggy wishes to commit to a bit (property (i)), she writes it on the floor and, before allowing Vic to look, she covers it with opaque tape. Although Vic cannot tell which bit is hidden under the tape (property (iii)), Peggy can no longer change it. To "open the blob" (property (ii)), Peggy allows Vic to strip off the tape and look at the bit. Property (iv) is satisfied provided that the way in which the bit is written on the floor, the tape, and its placement are all uncorrelated to any secret Peggy wishes to keep from Vic.

In the following sections, we assume that blobs are available and show how to use them to obtain general minimum disclosure protocols. Sections 2 and 3 deal with the case of deterministically verifiable information. After a complexity-theoretic interlude in Section 4, Section 5 gives the general protocol for probabilistically verifiable information. Under various assumptions, Section 6 describes several implementations for blobs and compares their relative strengths and weaknesses. As we shall see, some blob implementations lead to protocols that protect Peggy's information unconditionally but that would allow her to lie to Vic by breaking some cryptographic assumption in real time. One subtle point—not illustrated by the floor-and-tape example—is that it is not necessarily the case that each given blob must encode a unique bit; more generally, it is Peggy's knowledge about the blob that determines which bit is involved. Dual blob implementations are unconditionally secure for Vic, but could allow him to recover Peggy's information after some long (perhaps infeasible) off-line computation. Other implementations show neither weakness, but rely on dogmas of quantum physics or require the participation of several parties. The last section compares these possibilities.

1.1. *Related work*

As occurs often in research, some of the ideas presented here were developed independently in several places. An early interactive proof was presented by Rabin [R2]. This concept was formalized and the notion of "zero-knowledge" protocols (which is related to minimum disclosure) were introduced in [GMR]. Also, [Ba] formalized a notion similar to that of interactive proofs. The model proposed in [GMR, Ba] is quite interesting from a theoretical point of view, but it is based on the assumption that the prover has unlimited computing power.

Assuming only the existence of secure probabilistic encryption schemes (in the sense of [GM]), [GMW] showed that "every language in NP has a zero-knowledge interactive proof system in which the prover is a probabilistic polynomial-time machine that gets an NP proof as an auxiliary input." Under a stronger assumption, the same result was obtained independently but subsequently in [BC1]. A similar result was also obtained independently by [Ch4], but in a very different model, which emphasizes the unconditional privacy of the prover's secret information, even if the verifier has unlimited computing resources. This model was set forward in [Ch2] and the result of [Ch4] is a special case of a protocol previously presented in [Ch1], whose properties are described in [Ch2, p. 1039]. (The results of [Ch4] (then [Ch3]) and [GMW] were first presented explicitly in March 1986 at the Marseille conference on algorithms, randomness and complexity.) Finally, [BC2] considered a model in which all parties involved are assumed to have "reasonable" computing power (this model is also compatible with the setting of [Ch4]). The current paper unifies all of these approaches.

The difference between these models can be illustrated by an example. Consider again the statement by which Peggy claims to know the prime factorization of some public integer n. In the [GMR] model, there would be no point for her to spend time convincing Vic of this, because Vic knows that it is an immediate consequence of her unlimited computing power. In the setting of [Ch4], her secret factorization cannot possibly be unconditionally secure once the integer n is made public; she may therefore just as well convince Vic that she knows the factors by giving them explicitly to him. (But if Peggy's statement had merely been that she knows non-trivial divisors of n, and if n is the product of several primes, the setting of [Ch4] would allow Peggy to convince Vic of her knowledge without disclosing any information as to which divisors she knows, even if Vic has unlimited computing power.) In the context of [BC2], on the other hand, it makes perfect sense for Peggy to wish to convince Vic of her knowledge via a protocol that discloses nothing that could help Vic compute the factors of n. In other words, the protocol is designed to make Vic's factoring task just as difficult after the protocol as it was before.

As we shall see in Section 7, it is also interesting to distinguish between the parties' available computing resources *during* and *after* the protocol. Our main result is a protocol that is unconditionally secure for *both* parties as long as Peggy is incapable of factoring a large integer (or extracting a discrete logarithm, or both simultaneously) *while the protocol is taking place*. Once the protocol is over, it is too late for either party to attempt any kind of cheating, regardless of their computing power. This is in sharp contrast with the result of [GMW, BC1] concerning NP-complete problems, which allows Vic to take as much time as he likes in attempts to extract Peggy's secret by deciphering the protocol's transcript off-line.

2. The Basic Protocol

Assume Peggy knows a satisfying assignment of truth values for some Boolean formula. The basic protocol allows Peggy to convince Vic that she knows such an assignment without revealing any information about it. This protocol follows the lines of [Ch4]. (Other constructions are given in [GMW, BC2], but [GMW] requires a reduction to a graph colouring problem and [BC2] requires that blobs satisfy additional properties.) As an example, consider the Boolean formula

$$\Psi = [(p \textbf{ and } q) \textbf{ xor } (\bar{q} \textbf{ or } r)] \textbf{ and } \overline{[(\bar{r} \textbf{ xor } q) \textbf{ or } (p \textbf{ and } \bar{r})]}$$

and let $\langle p = true,\ q = false,\ r = true\rangle$ be Peggy's secret satisfying assignment. (This is of course a toy example, since it would be too easy for Vic or anyone else to find out how to satisfy such a simple Boolean formula.)

As a first step, Peggy and Vic agree on the layout of a Boolean circuit to compute Ψ. For simplicity, we use only basic binary gates and negations in the circuit. (Of course, negations are not needed, since any Boolean formula can be rewritten efficienctly using only "NAND" gates.) The circuit for Ψ is illustrated in Fig. 1. In addition, this figure shows Peggy's satisfying assignment and the truth table of each gate (except the negation gates). Observe that one row is outlined in each truth table, corresponding to the circuit's computation on Peggy's satisfying assignment. Seeing the rows outlined is enough to easily verify that Ψ is satisfiable. This is achieved by simple independent checks on the consistency of each wire. For instance, the output of the top left "AND" gate is 0, which is indeed the first input of the middle row "EXCLUSIVE-OR" gate. Also, the first inputs to the top left and top right "AND" gates are the same, as they should be since they correspond to the same input variable. Lastly, the output of the final gate is 1. Notice that seeing these outlined rows also gives away the corresponding satisfying assignment (even if it were not written explicitly). The basic protocol allows Peggy to convince Vic that she knows how to so outline one row in each truth table—without revealing any information about which rows they are.

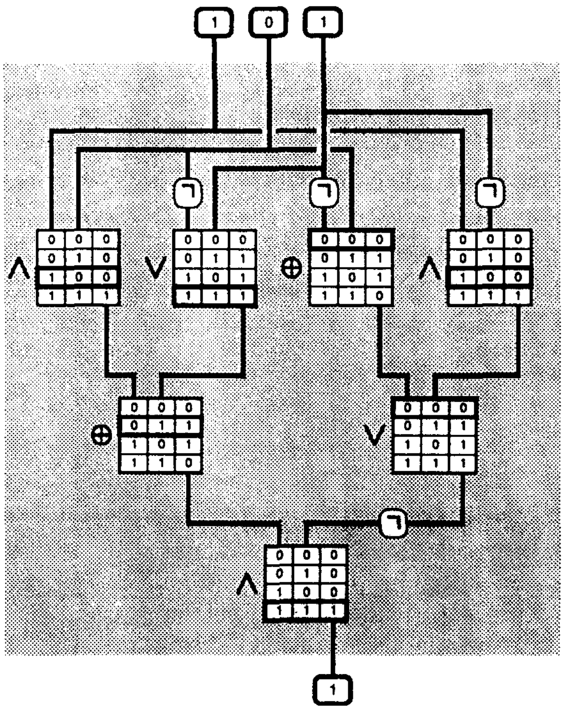

FIG. 1. A Boolean circuit with explicit truth tables and rows outlined.

This is achieved by an interactive protocol consisting of several rounds. In each round, Peggy "scrambles" the circuit's truth tables and commits to a corresponding collection of blobs. At this point, Vic issues one of two possible challenges to Peggy: one challenge requires Peggy to show that the blobs really encode a valid scrambling of the circuit's truth tables; the other challenge requires Peggy to open the rows that would be outlined, assuming it is a valid scrambling. The challenges are thus designed in such a way that Peggy could meet *both* of them only if she knew how to satisfy the circuit, but answering either *one* of them yields no information about how. Because Peggy cannot predict ahead of time which challenges will be issued by Vic, each round increases Vic's confidence in Peggy. In fact, Peggy would be caught cheating with probability at least 50% in each round if she were not able to answer both possible challenges, so that she could only hope to fool Vic in k successive rounds with exponentially vanishing probability 2^{-k}.

We call such techniques "cut-and-choose" because each round is similar to the classic "protocol" by which two children split a piece of cake—one of them cuts and the other one chooses. The great utility of a cut-and-choose like ours is that it gives an exponential increase in security at the cost of only a linear increase in the number of rounds. The earliest use of such cut-and-choose that we know of in the context of cryptographic protocols was presented by Rabin in 1977 [R2].

The "scrambling" of each truth table by Peggy consists of a random row permutation and column complementation. Let us illustrate this principle with an example. Figure 2a shows the truth table for the Boolean conjunction ("AND"). The rows of this table are randomly permuted to yield the table given in Fig. 2b. (Each of the 24 possible permutations—including the identity permutation—may be chosen with uniform probability.) Then, one bit is randomly chosen for each of the three columns of the truth table. Finally, each column is complemented if and only if its corresponding random bit is a 1, as shown in the three intervening tables. The final result is illustrated in Fig. 2c. Notice that the whole scrambled truth table can still unmistakably be recognized as representing the Boolean conjunction (provided the complementation bits, shown within circles throughout the drawings, are specified).

The complementations must be chosen consistently; all truth table columns

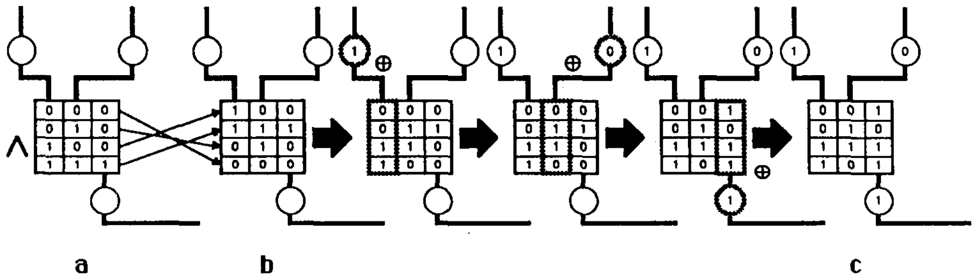

FIG. 2. Permutation and complementation of a truth table.

corresponding to the same wire in the circuit must either all be complemented or all remain the same. This is achieved by choosing randomly and independently the complementation bits corresponding to each wire. (For simplicity, we never complement the output of the final gate.) Figure 3 gives the result of random permutations and complementations of the truth tables in our original circuit from Fig. 1.

After producing a circuit similar to that of Fig. 3, Peggy commits to it: for each truth-table bit, Peggy commits to a blob that she knows how to open accordingly. (It is not necessary for her to actually commit to the complementation bits, but they must remain secret for the moment.) Coming back to our previous example blobs, one may think of Peggy having drawn Fig. 3 on the floor but having covered its bits with opaque tape before allowing Vic to look. Now that the "cut" is completed by Peggy, it is time for Vic to "choose": Vic asks Peggy to convince him of her good faith by requesting that she answers, at his random choice, either challenge "A" or challenge "B", defined as follows:

• If the challenge is "A," Peggy must open each and every blob she just committed to. Moreover, she must also reveal all the complementation bits that she used in the scrambling process. Continuing our intuitive image, Peggy strips off all the tape in order to show Vic the equivalent of Fig. 3. This allows Vic to verify that

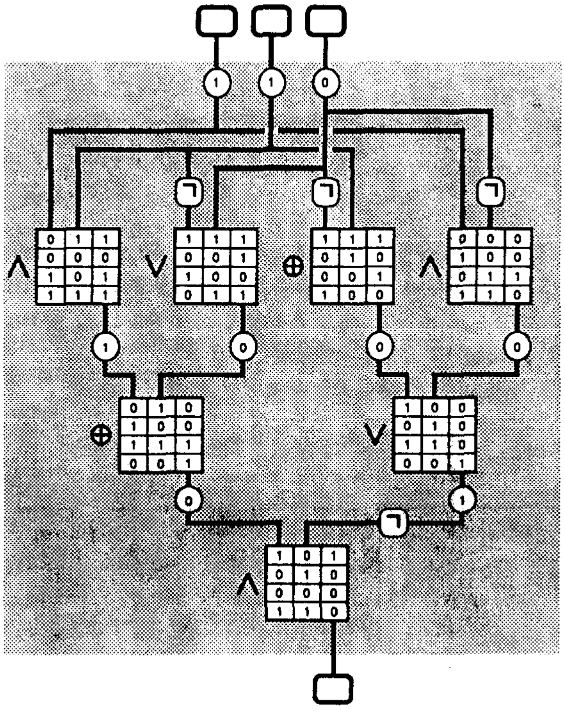

FIG. 3. A circuit with randomly permutted and complemented truth tables.

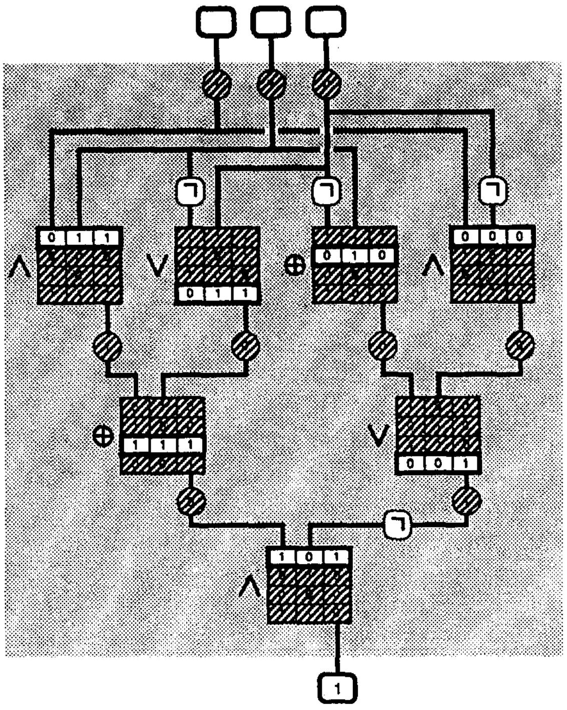

FIG. 4. The existence of a satisfying assignment is shown.

the information concealed by the blobs corresponds to valid permutations and complementations of the Boolean circuit's truth tables.

• If the challenge is "B," Peggy opens only the blobs corresponding to one row in each truth table. The rows to be opened are precisely those that were outlined in Fig. 1 in their (probably) new location determined by the row permutations. Still continuing our image, Peggy selectively strips away pieces of tape in order to show Vic the equivalent of Fig. 4. This allows Vic to verify the consistency of each wire and the fact that the final output of the circuit is a 1 bit.

3. PROOF OF THE BASIC PROTOCOL

Three requirements must be satisfied in order to prove correctness of the basic protocol; the following must hold, except perhaps with an exponentially small probability:

(1) Peggy can carry out her share of the protocol, provided she knows a satisfying assignment for Ψ. (Of course, no protocol could possibly *force* Vic to be convinced, even giving him the satisfying assignment in the clear, because he can always refuse to listen.)

(2) If Peggy does not know a satisfying assignment for Ψ, no matter how she pretends to follow the protocol, Vic will catch her cheating.

(3) If Peggy knows a satisfying assignment for Ψ and if she faithfully follows her share of the protocol, she does not reveal anything to Vic that could help him determine her satisfying assignment (or even find partial information about it)—this remains true even if Vic deviates arbitrarily from his stipulated behaviour in the protocol.

Defining properties (i) and (ii) of blobs allow Peggy to commit to blobs and open them as needed. Also, anyone can randomly permute the rows and complement the columns of a truth table, and thus obtain the equivalent of Fig. 3. Knowing a satisfying assignment for Ψ allows Peggy to outline one row in each of the scrambled truth tables. She can do this simply by remembering which columns are complemented and where the random truth-table permutations have taken each row that she knows would be outlined in the originally agreed circuit. Thus, the first requirement is satisfied.

The second requirement is satisfied because of the bit commitment property (ii) of the blobs. Assume Peggy does not know how to satisfy Ψ. In any given round, she can either commit to genuine permutations and complementations of the original circuit's truth tables (similar to Fig. 3), or she can commit to something phoney. In the first case, she cannot meet challenge "B" without knowing a satisfying assignment for Ψ; in the second case, she cannot meet challenge "A" without breaking the bit commitment property of blobs. Therefore, as long as she cannot predict the challenges to be issued by Vic, she has at least a 50% chance of being caught in each round. As mentioned earlier, her probability of fooling Vic in k successive rounds is therefore at best 2^{-k}.

The reason why the third requirement is satisfied is more subtle. Let us first argue that Vic cannot learn anything about the satisfying assignment (beyond the fact that Peggy knows it) from seeing either Fig. 3 or Fig. 4 alone. If he issues challenge "A" and thus gets to see Fig. 3, he obtains randomly permuted and complemented versions of the agreed circuit's truth tables. This is of no possible use to Vic, because he could have produced such a figure just as well by himself (even if the Boolean formula were *not* satisfiable). On the other hand, if he issues challenge "B" in order to see Fig. 4, what he gets amounts to the result of applying a true one-time pad (Peggy's independent random complementations) on the Boolean values carried by the circuit's wires while it computes a satisfying assignment (except for the final output wire, which should carry the value 1). Since such a one-time pad hides all the information, this gives no advantage either in finding the assignment. In other words, it is only by matching a Fig. 4 with its corresponding Fig. 3 that would allow Vic to learn something about the satisfying assignment; but of course Peggy will never answer both challenges in the same round.

Thanks to the defining property (iii) of blobs, Vic is prevented from matching any Fig. 4 he can get by issuing challenge "B" with its corresponding Fig. 3. (Property (iii) is not relevant when Vic issues challenge "A", since in this case

Peggy opens all the blobs involved in the corresponding round.) Finally, property (iv) prevents Vic from learning anything about Peggy's secret satisfying assignment from the processes by which Peggy commits to and opens blobs.

Even though the third requirement is satisfied, this does *not*, in general, imply that Vic cannot obtain anything beyond the fact that Peggy genuinely knows a satisfying assignment for Ψ. For instance, it is possible that only Peggy has the technology or knowledge necessary to commit to these blobs, in which case Vic might obtain something he could not have produced himself—although not the satisfying assignment. A more interesting situation occurs if one considers a variation on the basic protocol in which all the rounds are carried out in parallel: Peggy commits all at once to blobs corresponding to k circuits similar to Fig. 3, Vic sends his string of challenges, and Peggy opens the blobs as requested by the challenges. (This would be more efficient in some settings.) The modified protocol makes it possible for Vic to choose his challenges as a function of the entire collection of blob commitments. Although this cannot provide him with any advantage in discovering Peggy's satisfying assignment, it might allow him to subsequently convince others that Ψ is satisfiable by showing them the transcript of his conversation with Peggy (see Section 6.1.2 for an example of this situation). In other words, the parallel version of the basic protocol remains minimum disclosure, but it may not be "zero-knowledge" in the terminology of [GMR].

Intuitively, a protocol is *zero-knowledge* if the third requirement is strengthened to the effect that Vic cannot obtain anything at all beyond learning that Peggy knows a satisfying assignment. More precisely, Vic must be able to simulate his entire conversation with Peggy without in reality ever talking to her. (Refer to [GMR] for a formal definition.) Nevertheless, our basic protocol is zero-knowledge provided that blob defining property (iv) is strengthened to make sure that Vic does not gain *anything* from the process by which Peggy commits to blobs and that he obtains *only* the intended bits from the process by which Peggy opens some of them. Following the proof techniques of [GMR], we say that the blobs are *simulatable* if, in addition to properties (i), (ii), and (iii), they satisfy

(iv′) Vic can simulate what he would have been provided in the process by which Peggy commits to blobs that she could open as 0 and to blobs that she could open as 1. He can also simulate the process by which she would open these blobs had she committed to them herself.

Note that this new property implies the original (iv), since Vic must be able to simulate the commitment and opening of blobs even if he does not know Peggy's secret. If simulatable blobs are used, it is easy for Vic to attempt simulating one round of the protocol without talking to Peggy, except that he will fail with probability 50%. In order to do this, Vic proceeds as follows:

- he flips a fair coin to decide whether he will be prepared to answer challenge "A" or challenge "B";

• he randomly generates a Fig. 3 or a Fig. 4, depending on the outcome of the coin flip;

• he uses property (iv′) to simulate Peggy committing to a sequence of blobs that he knows how she would proceed to open in order to show whichever figure (3 or 4) he has just prepared.

• he then (honestly!) asks himself which challenge he would issue at this point if he had just received all these blob commitments from the real Peggy; and

• if he issued himself the challenge he can meet, he simulates Peggy opening the relevant blobs—otherwise he fails.

The crucial point is that the defining property (iii) of blobs ensures that there is no correlation between the challenge Vic decides he would be ready to meet and the challenge he actually issues to himself. In order to simulate the whole k-round protocol, Vic must repeat the above an average of $2k$ times, pretending that the unlucky rounds never happened.

This reasoning does *not* extend, in general, to the parallel version of the protocol. Assume for simplicity that blobs are bit strings and that Peggy commits to a blob by showing it in the clear. Consider the following stategy for Vic: after receiving blobs corresponding to all k circuits with randomly permuted and complemented truth tables from Peggy, he concatenates these blobs and uses the result as input to some one-way function. He then uses the first k bits of the output of this function to determine the k challenges to be issued. If Vic tries to adapt the above technique directly in order to simulate this protocol, the one-way function creates a dependency between the challenges that he is ready to meet and those that he actually issues to himself, resulting in an exponentially small probability of success. Even though running the protocol with Peggy does not help Vic in learning anything about Peggy's secret, its transcript may enable him to convince someone else of the existence of this secret, because Vic could almost certainly not have produced the transcript otherwise. This leads to a curious penomenon: the transcript of a parallel version of the protocol may contain no information on Peggy's secret (in the sense of Shannon's information theory [S]), yet it can be used to convince someone else of the secret's existence! In other words, the parallel version of the protocol is minimum disclosure, but it may not be zero-knowledge even if simulatable blobs are used.

If it is important that the protocol be carried out in parallel (perhaps for reasons of efficiency), the protocol remains zero-knowledge provided that defining property (iv) is strengthened further. We say that the blobs are *chameleon* if, in addition to properties (i), (ii), and (iii), they satisfy:

(iv″) Vic can simulate what he would have seen in the process by which Peggy commits to blobs. Moreover, for each of these blobs, Vic can simulate both the process by which Peggy would open it as a 0 and the process by which she would open it as a 1.

In other words, chameleon blobs allow Vic to do just what property (ii) prevents

Peggy from doing. Even if Peggy and Vic have similar computing abilities, as we shall see in Section 6.1, this property can sometimes be achieved if Vic has additional information. The advantage of chameleon blobs is that they allow Vic to simulate in a straightforward way his entire conversation with Peggy, without ever encountering failures. Again, this remains true even if Vic deviates arbitrarily from his stipulated behaviour. In our context, however, there is only one way in which Vic can deviate without Peggy stopping the protocol altogether: by choosing which challenges to issue in a way that depends on Peggy's blob commitments rather than choosing them randomly. However, no such strategy is of any use to Vic with chameleon blobs.

In order to simulate the parallel version of the protocol, Vic simulates Peggy's commitment to as many blobs as she would use. Because the blobs are chameleon, Vic does not need to have already decided in which way he expects Peggy to be able to open them. At this point, Vic looks at these commitments and chooses his challenges exactly as if the commitments actually came from Peggy. Whenever he chooses challenge "A," he randomly permutes and complements the Boolean circuit's truth tables to produce something like Fig. 3, and he "opens" all the corresponding blobs accordingly. Whenever he chooses challenge "B," he randomly selects one row in each truth table and one Boolean value for each wire in the circuit (except that he always selects 1 for the value of the final output wire); he then "opens" the blobs in these rows to reflect the value chosen for the corresponding wires, thus producing something like Fig. 4.

4. A Complexity-Theoretic Point of View

Because satisfiability of Boolean formulas is NP-complete [Co, GJ], the basic protocol can be used to supply minimum disclosure proofs of knowledge for any positive statement concerning a language L in NP. Assume without loss of generality that $L \subseteq \Sigma^*$, where Σ stands for $\{0, 1\}$ (i.e., elements of L consist of binary strings). By the definition of NP, there exists a "proof system" $Q \subseteq L \times \Sigma^*$ such that

- $(\forall x \in L)(\exists c \in \Sigma^*)[|c| \leqslant p(|x|)$ and $\langle x, c \rangle \in Q]$ for some fixed polynomial p, where $|x|$ denotes the length of x; and

- there exists a polynomial-time (deterministic) algorithm capable of deciding, given x and c, whether $\langle x, c \rangle \in Q$.

In other words, whenever $x \in L$, there exists a succinct "certificate" c to this effect, and one can efficiently verify that c is a valid proof that $x \in L$. Using our terminology, such a c is what we called "verifiable information" to the effect that $x \in L$.

Using Cook's theorem [Co], both Peggy and Vic can efficiently build from any $x \in \Sigma^*$ a Boolean formula $\Psi_L(x)$ satisfiable if and only if $x \in L$. Moreover, because the proof of Cook's theorem is constructive, it is enough for Peggy to know some succinct c such that $\langle x, c \rangle \in Q$ in order to efficiently deduce a satisfying assignment for $\Psi_L(x)$.

Therefore, if $L \in \mathrm{NP}$, $x \in L$, and if Peggy knows a succinct certificate c to the effect that $x \in L$, then Peggy can use the basic protocol to convince Vic that $\Psi_L(x)$ is satisfiable, hence that $x \in L$ and that she knows how to prove it. This is a minimum disclosure protocol, assuming of course that Vic already knows the proof system for L and our basic protocol (otherwise, much information is given to Vic when Peggy instructs him about these). For most practical applications, it is better to think of an *ad hoc* verifying circuit, rather than building it through the machinery of Cook's theorem.

As pointed out by [FFS], one may prefer not to call this type of protocol "*zero*-knowledge" because Vic does gain knowledge from running it—in particular, he learns that $x \in L$. Following [GHY], this is why we use the word "minimum" rather than "zero": Vic learns that $x \in L$, as intended, but not the proof of this fact. We use "disclosure" rather that "knowledge" because Vic may gain additional knowledge, in general, if the blobs are not simulatable or if the protocol is carried out in parallel.

It is interesting to consider both restrictions and extensions of NP in the context of minimum disclosure proofs of knowledge.

The interesting restriction concerns languages $L \in \mathrm{NP} \cap \text{co-NP}$. In this case, one can construct for each $x \in \Sigma^*$ two Boolean formulae $A_L(x)$ and $B_L(x)$ such that exactly one of them is satisfiable ($A_L(x)$ if $x \in L$ and $B_L(x)$ if $x \notin L$). Clearly, their disjunction $C_L(x) = [A_L(x) \textbf{ or } B_L(x)]$ is always satisfiable. Assume now that Peggy knows whether $x \in L$ or not, and that she has the corresponding succinct NP certificate. This gives her a satisfying assignment for either $A_L(x)$ or $B_L(x)$, whichever is satisfiable; hence she can also satisfy $C_L(x)$. Consider what happens if she uses our basic protocol to convince Vic that she knows a satisfying assignment for $C_L(x)$. Clearly, this does not disclose anything about x to Vic (because $C_L(x)$ is always satisfiable). However, it convinces Vic that Peggy knows whether $x \in L$ or not, and that she can prove it. This issue and its applications to identification systems are discussed in [FFS], but such systems must be used with caution because they are not always as secure as they may seem [BBDGQ].

Minimum disclosure protocols can also extend beyond NP if we allow the certifying procedure to be probabilistic. Recall that BPP stands for the class of decision problems that can be solved in probabilistic polynomial time with bounded error probability [G]. It is reasonable to consider BPP as the *real* class of tractable problems (rather than P) because the error probability can always be decreased below any threshold $\delta > 0$ by repeating the algorithm $\alpha \log \delta^{-1}$ times and taking the majority answer, where the constant α depends only on the original error probability [BB]. It is generally believed that there is no inclusion relation either way between NP and BPP: non-determinism and randomness seem to be uncomparable powers. These powers can be combined in several ways. We consider Babai's class MA [Ba] to be the most natural, but we prefer calling it NBPP. This class is such that $\mathrm{NP} \cup \mathrm{BPP} \subseteq \mathrm{NBPP}$, hence NP is almost certainly a strict subset of NBPP.

The class NBPP is defined exactly as NP, except that we are satisfied with a BPP algorithm for deciding, given x and c, whether $\langle x, c\rangle \in Q$ (i.e., we only require that $Q \in \mathrm{BPP}$). Whenever $\langle x, c\rangle \in Q$, we now refer to c as a *convincing argument* for the fact that $x \in L$ (we no longer call it a *certificate* because it cannot be verified with certainty, in general).

Section 5 shows how to obtain nearly minimum disclosure protocols for any language L in NBPP. As is usual in this paper, we assume that both Peggy and Vic have "reasonable" computing power and similar algorithmic knowledge, but that Peggy is initially given a succinct convincing argument c to the effect that $x \in L$. This allows Peggy to initially convince herself beyond any reasonable doubt that $x \in L$, by running the BPP algorithm on input $\langle x, c\rangle$. The purpose of our protocol is for Peggy to convince Vic that she knows such a c, without disclosing anything that could help him find it.

The class NBPP is Babai's class MA, which he defined for his "Arthur–Merlin games" [Ba] (and is similar to Papadimitriou's "stochastic satisfiability" in his "games against nature" [Pa]). According to Babai, his other class AM is a better candidate for the generalization of NP to probabilistic computations. In particular, he proved that $\mathrm{MA} \subseteq \mathrm{AM}$. The interest in AM is further increased by the proof that, for any fixed $k \geqslant 2$, $\mathrm{AM} = \mathrm{IP}(k)$, the class of languages that allow an interactive protocol with no more than k rounds [GS]. All these considerations are theoretically very compelling.

We claim nonetheless that MA (i.e., NBPP) is a more natural class for *practical* purposes, at least in cryptographic settings. If $L \in \mathrm{NBPP}$ and $x \in L$, it is enough for Peggy to know one succinct convincing argument c to this effect. In practice, this c need not be God-given to Peggy. As mentioned earlier, it is easy for some languages to generate both x and the corresponding c by a probabilistic process. Consider, for instance, the set B of integers having exactly two prime factors. If Peggy generates two distinct random integers p and q that pass a probabilistic primality test [SS, R1] to her satisfaction, she is convinced that $n = pq$ is a member of B and her convincing argument is $\langle p, q\rangle$. The protocol given in Section 5 allows her to convince Vic that $n \in B$ without disclosing anything else that might help him factor n. (In this case, $B \in \mathrm{NP}$ because the set of primes is in NP [Pr]. This does not reduce the practical interest of our example, however, because Peggy may find it prohibitive to convert her NBPP convincing argument $\langle p, q\rangle$ into an NP certificate $\langle p, c(p), q, c(q)\rangle$, where $c(\bullet)$ stands for an NP certificate that $\bullet$ is prime. This remark remains true in practice despite the results of [GK, AH].)

By contrast, it is not clear that Peggy can reasonably be asked to carry out an AM protocol—regardless of the minimum disclosure considerations—even if she were initially given a succinct piece of advice: an AM protocol would, in general, require Peggy to determine an NP-like certificate as a function of a random string supplied by Vic. This is a pity in a way, because it is trivial that our basic protocol allows the transformation of any AM protocol into a minimum disclosure one. As explained in [GMW], this is true because once "Arthur" has given "Merlin" his coin flips, it "only" remains for Merlin to satisfy an NP statement, which can be done without disclosing anything else if the basic protocol of Section 2 is used with simulatable blobs.

5. Going Beyond NP: The Probabilistic Case

Consider any language $L \in \mathrm{NBPP}$. Let x be such that Peggy knows a succinct convincing argument c to the effect that $x \in L$. Because c is not an NP certificate, Peggy is not absolutely certain that $x \in L$, but she can reduce her probability of error below any desired threshold by the virtues of BPP. The purpose of the minimum disclosure protocol described in this section is for Peggy to convince Vic that $x \in L$ and that she knows a convincing argument to this effect, but in a way that does not help Vic determine this convincing argument or any information about it. If the underlying blobs are simulatable, we call this process a "*non-transitive transfer of confidence*" because it convinces Vic that $x \in L$ (a statement about which Peggy is convinced already) in a way that he cannot subsequently convince anyone else.

Note that Vic could be fooled by this process in several different ways. It may be that Peggy is dishonest and that she does not really know a convincing argument to the effect that $x \in L$, but that she succeeds (with exponentially small probability) in fooling Vic by being lucky enough each time to be asked the only challenge she is capable of answering (exactly as she could have done with the basic deterministic protocol). It is also possible that Peggy is honest but wrong in her belief (because the certifying BPP algorithm misled her). In this case, it is most likely that Peggy will discover her mistake as a result of trying to convince Vic, but it is also possible that the certifying algorithm will err once more. Finally, it is possible that Peggy is honest and correct in her claim, but that when she runs the protocol with Vic, the verdict comes out wrong owing to an error of the certifying algorithm.

As a preliminary step, Peggy and Vic agree on the error probability δ they are willing to tolerate for the certifying algorithm. From this agreement, they modify the algorithm so that its probability of error does not exceed δ (for this, they first determine how many times the original algorithm must be repeated for the majority answer to be almost certainly correct [BB]). From now on, we assume without loss of generality that the probability of error of the certifying algorithm is negligible.

In essence, Peggy wants to convince Vic that she knows some secret input c such that the certifying algorithm will (almost certainly) accept the input $\langle x, c \rangle$. Let n and m denote the size of x and c, respectively. Assume for simplicity, and without loss of generality, that the value of m is uniquely determined as a known (easy to compute) function of n, so that the protocol need not hide the value of m from Vic. Let r be an upper bound on the number of coin flips that the certifying algorithm can perform on any input $\langle x, \hat{c} \rangle$, where $\hat{c}$ is of size m. An argument similar to the proof of Cook's theorem shows that this gives rise to a Boolean formula Ψ with at least $m+r$ variables. If the first m variables of this formula are set to represent the binary string c, and the next r variables are determined by independent random coin flips, then (except with probability at most δ) it is easy to set the other variables (if any) so as to satisfy the whole formula if and only if c is a valid convincing argument that $x \in L$. This formula can be constructed from knowledge of x and of the certifying procedure, with no need for the secret convincing argument c. Hence, it can be made public. As before, it is converted into a Boolean circuit on which both Peggy and Vic agree.

The basic minimum disclosure protocol of Section 2 cannot be used directly, because Vic cannot trust Peggy to choose the r appropriate inputs truly at random. On the other hand, Peggy cannot allow Vic to choose these variables either, because a careful choice might allow Vic to obtain information on Peggy's secret convincing argument c. It is therefore necessary to use a coin-flipping subprotocol to set these inputs to random values not under the control of either party. Moreover, Vic should not be allowed to see the outcome of the coin flips, again to prevent him from learning information about c (i.e., coin flipping should be performed "in a well" [Bl]). In order to allow Peggy to use these coin flips without ever showing their outcome to Vic, it is necessary that the coin-flipping protocol produce blob commitments rather than simply bits. Finally, Peggy must not be allowed to choose her c as a function of the coin flips.

It is much easier to implement all these requirements if we ask that blobs have two additional properties:

(v) Given two unopened blobs that she has committed to, Peggy can convince Vic that she could open them to show the same bit (provided this is so) without disclosing any additional information.

(vi) Given two unopened blobs that she has committed to, Peggy can convince Vic that she could open them to show distinct bits (provided this is so) without disclosing anything else.

As we shall see in Section 6.5, however, property (vi) is always a consequence of properties (i) through (v), and property (v) is itself a consequence of properties (i) through (iv).

Coin flipping capable of producing a blob is trivial to implement with property (vi): Peggy commits to two blobs that she can open to show distinct bits, she convinces Vic that this is so, and she asks him to choose one of them. When Vic makes his choice, the coin flip is determined and Peggy knows its outcome—which is the bit she could show by opening the blob chosen by Vic. However, Vic cannot tell how it went unless Peggy subsequently opens this blob (which she will *never* do in the protocol below). Property (iii) prevents Vic from influencing the coin flip, and properties (ii) and (vi) prevent Peggy from doing so.

We are now ready to describe our general protocol for the case of probabilistically verifiable information. Recall that Peggy and Vic have agreed on a Boolean circuit corresponding to the certifying algorithm intended to probabilistically verify

Peggy's secret convincing argument c that $x \in L$. At this point, Peggy commits once and for all to her convincing argument by committing to m blobs that she could open to show the bits of c. Then, Peggy and Vic flip r coins "in a well" by the above procedure, which results in r blob commitments corresponding to the outcome of the coin flips. Peggy can now use a slight variation on the basic protocol of Section 2 in order to convince Vic that she knows how to select the other inputs of the circuit (if any) so as to satisfy it.

The basic protocol must be modified in order to force Peggy to use the proper bits for the inputs corresponding to c and to the coin flips, but this must be achieved without disclosing anything that could help Vic learn about the value of these bits. We illustrate how this can be done with the example of Section 2. Assume that Peggy has committed to some blob b (which she could open as 1, but Vic does not know this). Peggy wishes to convince Vic that she knows a satisfying assignment for Ψ in which the first input corresponds to the bit she could open as blob b. For this purpose, Peggy randomly permutes and complements the Boolean circuit's truth tables to produce a Fig. 3, and she commits to it exactly as before. Moreover, for each input bit that Peggy has committed to (the first input bit in our example, the first $m+r$ bits in general), she now commits to the complementation bit used on the corresponding wire to produce the current Fig. 3.

If Vic issues challenge "A," Peggy opens each and every blob she has just committed to and reveals all the complementation bits used, thus showing Fig. 3 to Vic, still exactly as in the deterministic basic protocol. If Vic issues challenge "B," however, she must do more than showing Fig. 4 to Vic (which would say nothing about the first Boolean variable of the satisfying assignment). Because the wire corresponding to the first input is set to 0 in Fig. 4 (as shown by the first bit in the outlined row of the top left truth table), Peggy uses property (v) to convince Vic that she could open the blob associated with the corresponding wire complementation in the same way as she could open blob b. If the wire corresponding to this input had been a 1, she would, of course, have used property (vi) instead.

This completes the description of our protocol for the case of probabilistically verifiable information. In general, however, it is *not* a minimum disclosure protocol from a theoretical point of view. The subtle difficulty is that different convincing arguments may cause the certifying procedure to fail with different probabilities. Because he is generally unable to predict the failure probability, Vic cannot simulate exactly the conversation that would take place if he were really talking to Peggy. Moreover, running the protocol an exponentially large number of times with Peggy could in principle allow a very powerful Vic to increase his chances to guess correctly Peggy's secret (by keeping a tally of how many times the protocol showed a failure of the certifying algorithm). For all practical purposes, however, this threat is of no consequence if δ is chosen small enough; the protocol can thus be used safely.

A variation on this scheme is *almost* always minimum disclosure, but it is usually more time consuming. To achieve this, the original BPP certifying algorithm has to be modified, by repeating it enough times and taking the majority, so that all but exponentially few of the random choices may cause it to give the wrong answer on even a single input. The fact that this is possible can be proven by a refinement of the proof that $\mathrm{MA} \subseteq \mathrm{AM}$ [Ba]. This allows use of the basic protocol from Section 2 almost directly, with no need for the coin tossing to be in a well or for blob properties (v) and (vi).

6. Blob Implementations

We have taken the existence of blobs for granted in the previous sections. Let us now see how they can be implemented in practice. This can be done in several ways. None of these implementations is ideal, however. The choice of implementation should be based on the particular requirements of the application. The safety of most of the following implementations depends on unproved assumptions about the computational difficulty of solving particular problems. Section 7 compares the advantages and drawbacks of these various approaches.

As an elementary (but probably not very secure) example, consider two isomorphic graphs G and H upon which Peggy and Vic agree. Assume that Peggy is convinced that they are isomorphic, but that she does not actually know an isomorphism between them. Suppose further that she is computationally incapable of finding such an isomorphism in a reasonable amount of time. (Let us postpone until Section 6.1.3 the question of how Peggy could be convinced that the graphs are isomorphic, without herself explicitly knowing an isomorphism.) In this setting, Peggy agrees with Vic that any graph for which she can show an isomorphism with G (resp. H) is a commitment to the bit 0 (resp. 1). Referring to the defining properties of blobs, property (i) holds because Peggy can commit to the bit 0 (resp. 1) by randomly permuting the vertices of G (resp. H) and showing the resulting graph—the blob—to Vic. In order to open a blob, it suffices for Peggy to show Vic the isomorphism she knows with G or H, whichever is the case. Property (ii) holds if and only if Peggy cannot find an isomorphism between G and H while the protocol is in progress. (More precisely, in order for Peggy to break property (ii), she must have obtained information that makes it easy for her to discover such an isomorphism.) Property (iii) holds unconditionally because blobs used by Peggy as commitments to 0 are information-theoretically indistinguishable from those used as commitments to 1. These blobs are simulatable because property (iv′) is satisfied: Vic does not need Peggy's help to permute randomly the vertices of G and H. Finally, these blobs are chameleon—property (iv″)—if and only if Vic knows an isomorphism between G and H.

As illustrated by this example, it is not the blob itself (some graph isomorphic to both G and H) that determines a bit, but rather Peggy's knowledge about it (the actual isomorphism known by Peggy between this graph and either G or H). Thus, many *bit commitment schemes* (but not those of Sections 6.3 and 6.4) consist of two sets, X and Y, together with an efficiently computable *verification function* $v: X \times Y \rightarrow \{0, 1, \bullet\}$, where "$\bullet$" stands for "undefined" (that is, this $y \in Y$ is irrelevant for this $x \in X$). In order to commit to bit $b \in \{0, 1\}$, Peggy chooses a pair $x \in X$ and $y \in Y$ such that $v(x, y) = b$. Here, x is the blob and y is Peggy's additional knowledge about it. The actual commitment occurs when Peggy shows x to Vic. In order to open the blob, Peggy shows y to Vic and lets him compute $v(x, y)$. For this to be efficient and secure, we need the following properties:

(i) Given $b \in \{0, 1\}$, Peggy can generate pairs $x \in X$ and $y \in Y$ such that $v(x, y) = b$.

(ii) Peggy cannot obtain any triple $x \in X$, $y_0 \in Y$, and $y_1 \in Y$ such that $v(x, y_0) = 0$ and $v(x, y_1) = 1$.

(iii) When Peggy gives Vic some $x \in X$, Vic cannot learn anything about whether Peggy also knows a $y \in Y$ such that $v(x, y) = 0$ or such that $v(x, y) = 1$.

(iv) The way in which Peggy chooses her pairs $\langle x, y \rangle$ satisfying property (i) is uncorrelated to any secret she wishes to keep from Vic.

These four requirements are slight restrictions on the corresponding defining blob properties of Section 1, which is why we continue to refer to them by the same symbols. If the blobs are to be simulatable, we must also require that:

(iv′) Given $b \in \{0, 1\}$, Vic also can generate pairs $x \in X$ and $y \in Y$ such that $v(x, y) = b$. Moreover, Vic can generate these pairs with the same probability distribution as Peggy would according to property (i).

In particular, it is sufficient for Vic to know the process by which Peggy generates blobs satisfying property (i) for these blobs to be simulatable. This will in fact be the case throughout Sections 6.1 and 6.2, but it will not be repeated there. Finally, the blobs are chameleon provided that:

(iv″) Vic can generate triples $x \in X$, $y_0 \in Y$, and $y_1 \in Y$ such that $v(x, y_0) = 0$ and $v(x, y_1) = 1$. Moreover, the pairs $\langle x, y_0 \rangle$ (resp. $\langle x, y_1 \rangle$) thus generated are obtained with the same probability distribution as the pairs $\langle x, y \rangle$ that Peggy would generate according to property (i) in order to commit to the bit 0 (resp. 1).

There is an apparent contradiction between properties (ii) and (iii). If there exist x, y_0, and y_1 such that $v(x, y_0) = 0$ and $v(x, y_1) = 1$, why should Peggy be unable to obtain them, and thus violate property (ii)? On the other hand, if each $x \in X$ unambiguously determines the only possible non-undefined value for $v(x, y)$, why should Vic not be able to determine this value upon seeing x, and thus violate property (iii)?

We offer several different ways to resolve this. Section 6.1 investigates blobs that are *unconditionally secure for Peggy* (such as the graph isomorphism implementation outlined above). In this case, property (iii) holds regardless of Vic's computing power. This is achieved by asking that, for every x in X, there must exist at least one y_0 and one y_1 in Y such that $v(x, y_0) = 0$ and $v(x, y_1) = 1$. Moreover, the probability that Peggy generates any given blob x satisfying property (i) must be the same whether she wishes to commit to 0 or to 1. These additional requirements clearly imply that Vic cannot learn anything about which way Peggy is able to open any unopened blob she has committed to. However, they also imply that Peggy could in principle violate property (ii); but our implementations are designed to make this computationally infeasible for her (under suitable assumptions). As a result, these blobs are ruled out by the [GMR] model because an infinitely powerful prover could always cheat them, but they fit within the models of [Ch4, BC2]. Some of these blobs are chameleon, which therefore allows the basic protocol to be zero-knowledge even if carried out in parallel, as explained in Section 3.

Section 6.2 investigates blobs that are *unconditionally secure for Vic*. In this case, property (ii) holds regardless of Peggy's computing power. This is achieved by asking that, for every x in X, there must *not* exist simultaneously a y_0 and a y_1 in Y such that $v(x, y_0)=0$ and $v(x, y_1)=1$. This additional requirement clearly implies that Peggy is irrevocably committed to a specific bit each time she utters a blob. However, it also implies that Vic could in principle violate property (iii); but our implementations are designed to make this computationally infeasible for him (again under suitable assumptions). These blobs lead to zero-knowledge interactive protocols in the sense of [GMR] and to protocols similar to those of [GMW, BC1]. Of course, none of these blobs are chameleon.

Section 6.3 considers blobs that are secure even if all parties have unlimited computing power. These blobs do not fit the mold of the verification function v described earlier. Blobs of Section 6.3.1 make use of quantum physical principles. Using these blobs, it is provably impossible for Vic to obtain any information on Peggy's secret (assuming that quantum physics is correct). Although quantum blobs could in principle be cheated by Peggy, this would require a technology far beyond any foreseeable future. Blobs of Section 6.3.2 can be used in a multiparty environment under the assumption that the honest participants outnumber the cheaters.

Section 6.4 describes some relation between blobs and conceptually simpler primitives such as Rabin's oblivious transfer. This allows us to give very weak assumptions for the existence of blobs. These blobs also do not fit the mold of verification functions. Depending on the underlying oblivious transfer capability, we obtain blobs unconditionally secure for Peggy, for Vic, or for both of them.

Given any two blobs Peggy has committed to, Section 6.5 shows how she can convince Vic that she can open them to show either the same bit or distinct bits, whichever is the case, without disclosing anything else. This possibility was used extensively in Section 5 (optional properties (v) and (vi)).

Finally, Section 6.6 outlines a potential generalization to multi-valued blobs, which allows the efficiency of the basic protocol of Section 2 to be improved.

6.1. *Blobs Unconditionally Secure for the Prover*

6.1.1. *Based on Factoring* [BC2]

Some elementary number theory is necessary to understand this particular implementation of blobs. Let n be an integer. $\mathbb{Z}_n^*$ denotes the set of integers relatively prime to n between 1 and $n-1$. An integer $x \in \mathbb{Z}_n^*$ is a *quadratic residue* modulo n if there exists a $y \in \mathbb{Z}_n^*$ such that $x \equiv y^2 \pmod{n}$. This is denoted as $x \in \mathrm{QR}_n$. Such a y is called a *square root* of x, modulo n. Let s be any fixed quadratic residue. A uniformly distributed random quadratic residue can be generated by choosing $y \in \mathbb{Z}_n^*$ at random and computing $x = y^2 s \bmod n$. This holds in particular if $s = 1$. The crucial fact is that it is *information-theoretically* impossible to distinguish a quadratic residue thus produced using any given $s \in \mathrm{QR}_n$ from one produced using $s = 1$.

Now, let $n = pq$ be the product of two distinct odd primes. The problem of extracting square roots modulo n is computationally equivalent to the problem of factoring n [R3]. We shall assume here that factoring n is almost always infeasible when p and q are sufficiently large. Therefore, given n and $s \in \mathrm{QR}_n$, we assume that it is infeasible to compute a square root of s modulo n unless the factorization of n is known.

At the outset of the protocol, Vic randomly chooses two distinct large primes p and q, and he computes their product $n = pq$. Vic also chooses a random $t \in \mathbb{Z}_n^*$ ($t \neq \pm 1$) and computes $s = t^2 \bmod n$. Vic gives n and s to Peggy. Using a minimum disclosure protocol [Be, BC1], Vic convinces Peggy that s is a quadratic residue modulo n and that he knows one of its square roots. (Notice that in this initialization phase of the protocol, Vic temporarily takes the role of prover and Peggy that of verifier.)

The blobs are now defined by the sets $X = \mathrm{QR}_n$, $Y = \mathbb{Z}_n^*$, and

$$v(x, y) = \begin{cases} 0 & \text{if } x \equiv y^2 \pmod{n} \\ 1 & \text{if } x \equiv y^2 s \pmod{n} \\ \bullet & \text{otherwise.} \end{cases}$$

Property (i) holds because whenever she wishes to commit to some bit b, Peggy randomly chooses a $y \in \mathbb{Z}_n^*$ and computes $x = y^2 s^b \bmod n$. She gives x to Vic but keeps y as her secret *witness* that allows her to open blob x as bit b. Clearly, any quadratic residue can be used by Peggy as a commitment to 0 just as well as to 1, depending only on her knowledge about it. Therefore, property (iii) holds in a very strong sense: blobs committed to by Peggy convey *no* information on the bits she could show by opening them. Property (ii) holds computationally because Peggy could easily obtain a square root of s (which we assumed to be infeasible for her) from knowledge of y_0 and y_1 such that $y_0^2 \equiv y_1^2 s \pmod{n}$.

It is obvious that these blobs leave the door wide open for Peggy to cheat if she succeeds in extracting a square root of s. There is also a subtler possibility for Vic to cheat and thus learn everything about Peggy's secret. In order to achieve this, Vic must be "daring" from the beginning, because he must give Peggy a quadratic *non*-residue as his s. If he succeeds in convincing Peggy that s is a quadratic residue—which can only happen with an exponentially small probability—then Peggy can open as 0 blobs she has committed to precisely if they are quadratic residues, a condition that Vic can easily determine with the help of his factorization of n. Nevertheless, we classify these blobs as "unconditionally secure for Peggy" because only luck can allow Vic to cheat—no amount of computing power can help him.

In addition, these blobs are chameleon because Vic's knowledge of t, a square root of s, allows him to create blobs for which he can simulate Peggy's opening as either 0 or 1. To do this, Vic generates a random $y \in \mathbb{Z}_n^*$ and computes $x = y^2 s \bmod n$ and $\hat{y} = yt \bmod n$. He can then simulate Peggy's opening of this blob as a 0 (resp. 1), by using $\hat{y}$ (resp. y).

6.1.2. *Based on the Discrete Logarithm* [CDG, BKK]

Let p be a large prime and let α generate $\mathbb{Z}_p^*$, the multiplicative group of integers modulo p. Given any integer y, it is easy to compute $\alpha^y \bmod p$, but no efficient algorithm is known to invert this process, an operation known as computing the "discrete logarithm modulo p." The intractability assumption of the discrete logarithm was used in the very first paper published on public-key cryptography [DH]. It can also be used to create blobs, provided it is strengthened to assume that computing discrete logarithms modulo a large prime p remains infeasible even if the factorization of $p-1$ is known.

At the outset of the protocol, Peggy and Vic agree on a prime number p for which both of them know the factorization of $p-1$. They also agree on α, a generator of the group $\mathbb{Z}_p^*$. Thanks to their knowledge of the factors of $p-1$, they can both verify with certainty that p is a prime and that α is a generator. These same parameters p and α can be public, in the sense that they can be used with no breach of security by all parties wishing to engage in minimum disclosure protocols. At the outset, Vic also chooses a random $s \in \mathbb{Z}_p^*$ $(s \neq 1)$ and gives it to Peggy. Assuming the intractability of the discrete logarithm, Peggy cannot compute e such that $s \equiv \alpha^e \pmod{p}$.

The blobs are now defined by the sets $X = \mathbb{Z}_p^*$, $Y = \{0, 1, 2, \ldots, p-2\}$, and

$$v(x, y) = \begin{cases} 0 & \text{if } x \equiv \alpha^y \pmod{p} \\ 1 & \text{if } x \equiv s\alpha^y \pmod{p} \\ \bullet & \text{otherwise.} \end{cases}$$

Property (i) holds because whenever she wishes to commit to some bit b, Peggy randomly chooses a $y \in Y$ and computes $x = s^b \alpha^y \bmod p$. She gives x to Vic but keeps y secret as her witness that allows her to open blob x as bit b. Clearly, any element of $\mathbb{Z}_p^*$ can be used by Peggy as commitment to 0 just as well as to 1, depending only on her knowledge about it. Therefore, property (iii) holds unconditionally, as with the implementation of Section 6.1.1: blobs committed to by Peggy still contain no information on the way in which she could open them. Property (ii) holds computationally because Peggy could easily obtain e (which we assumed to be infeasible for her) from knowledge of y_0 and y_1 such that $\alpha^{y_1} \equiv s\alpha^{y_2} \pmod{p}$.

Despite a superficial resemblance, there is a fundamental difference between this implementation of blobs and that of Section 6.1.1. There is no longer any possibility for Vic to cheat. The fact that blobs that Peggy can open as 0 and blobs that she can open as 1 are information-theoretically indistinguishable depends only on the fact that p is a prime and that α generates $\mathbb{Z}_p^*$, and both are verifiable by Peggy before starting the protocol. Using the terminology of [GMW], this implementation of blobs turns the basic protocol presented here into a "perfect zero-knowledge interactive protocol" for satisfiability (except that it does not fit their model as an interactive protocol, since they allow the prover to be infinitely powerful, in which case she would have no problem computing e—which explains why Fortnow's theorem [F] does not apply). Such a perfect zero-knowledge interactive protocol was *incorrectly* claimed in [BC2] of the implementation corresponding to the blobs of Section 6.1.1. Notice, however, that it is computationally more efficient to use the blobs of Section 6.1.1.

Besides efficiency, there is another price to pay for making it impossible for Vic to cheat: the "discrete logarithm blobs" are not chameleon, and thus the basic protocol should not immediately be performed in parallel if it is to be zero-knowledge. If it were performed in parallel, Vic could cheat by choosing a random integer e and computing $2\alpha^e \bmod p$ as the s he gives to Peggy. Assume that Peggy uses the parallel version of the protocol to convince Vic that she knows the proof of some theorem **T**. If Vic uses a one-way function, for example, to select his challenges, he could subsequently use the transcript, together with the value of e, to convince others that **T** is true. Indeed, there is no obvious way in which Vic could have created this transcript by himself, unless he knows a proof of **T** or the discrete logarithm of 2. This illustrates a very curious phenomenon: although the transcript of the protocol can be used as evidence that **T** is true, it cannot be used in any way to facilitate finding such a proof. Moreover, the transcript contains no information on the proof of **T**, even in the sense of Shannon's information theory [S]!

With some preprocessing, it is possible to add the chameleon property to these blobs. Rather than choosing s randomly in $\mathbb{Z}_p^*$, Vic randomly chooses an integer e between 1 and $p-2$ and computes $s = \alpha^e \bmod p$. Using a minimum disclosure protocol [CEGP], he then convinces Peggy that he knows the discrete logarithm of s, which is all he needs to meet property (iv″). Note that in this case Vic would also convince Peggy that s is in the subgroup generated by α, so that the requirement that α be a generator of $\mathbb{Z}_p^*$ is no longer crucial for Peggy's safety. Therefore, if we tolerate an exponentially small probability that Vic could gain information on Peggy's secret, the factorization of $p-1$ need not be known and thus the assumption about the difficulty of computing discrete logarithms can be relaxed.

6.1.3. *Based on Graph Isomorphism* [BC2]

Define a graph G to be *hard* if it is computationally infeasible with high probability, given G and a random isomorphic copy of G, to figure out the isomorphism. We assume that hard graphs exist and that they can be obtained in practice.

At the outset of the protocol, Peggy and Vic agree on some hard graph $G = \langle N, E \rangle$. Vic randomly chooses a permutation $\sigma: N \to N$ and uses it to produce

$H = \langle N, E' \rangle$, where $(u, v) \in E'$ if and only if $(\sigma(u), \sigma(v)) \in E$. He then gives H to Peggy and convinces her that G and H are isomorphic without disclosing anything about the isomorphism σ [GMW]. By our assumption, it is computationally infeasible for Peggy to determine σ (or any other isomorphism between G and H) in a reasonable amount of time.

The blobs are now defined by the sets $X = \{K = \langle N, \hat{E} \rangle \mid K \text{ is a graph isomorphic to } G\}$, $Y = \{\gamma : N \to N \mid \gamma \text{ is a permutation}\}$, and

$$v(\langle N, \hat{E} \rangle, \gamma) = \begin{cases} 0 & \text{if } (u, v) \in \hat{E} \text{ iff } (\gamma(u), \gamma(v)) \in E \\ 1 & \text{if } (u, v) \in \hat{E} \text{ iff } (\gamma(u), \gamma(v)) \in E' \\ \bullet & \text{otherwise.} \end{cases}$$

The reader can easily verify that all the defining blob properties hold and that these blobs are also chameleon.

6.2. *Blobs Unconditionally Secure for the Verifier*

6.2.1. *Based on Quadratic Residuosity* [BC1]

To understand these blobs, further elementary number theory is needed. We refer the reader to [BC1] for the relevant background.

At the outset of the protocol, Peggy randomly chooses two distinct large primes p and q, and she computes their product $n = pq$. She also randomly chooses a quadratic non-residue s modulo n with Jacobi symbol $+1$. She discloses n and s to Vic. She then convinces Vic that n has only two prime factors [PG] and that s is a quadratic non-residue modulo n [GMR, GHY], without disclosing any additional information. Following the quadratic residuosity assumption [GM], we assume that Vic cannot distinguish random quadratic residues from non-residues with Jacobi symbol $+1$.

The blobs are now defined by the sets $X = \mathbb{Z}_n^*[+1]$, $Y = \mathbb{Z}_n^*$, and

$$v(x, y) = \begin{cases} 0 & \text{if } x \equiv y^2 \pmod{n} \\ 1 & \text{if } x \equiv y^2 s \pmod{n} \\ \bullet & \text{otherwise.} \end{cases}$$

Property (i) holds because whenever she wishes to commit to some bit b, Peggy randomly chooses a $y \in \mathbb{Z}_n^*$ and computes $x = y^2 s^b \bmod n$. She gives x to Vic but keeps y secret as her witness that allows her to open blob x as bit b. (Although Peggy's knowledge of the factors of n would allow her to recompute y from x, she saves time and effort by remembering it.) Peggy can open x as 0 if and only if it is a quadratic residue. This shows that property (ii) holds unconditionally, because any given x is either a quadratic residue or not. Property (iii), however, holds only computationally, because we have assumed that testing quadratic residuosity is infeasible for Vic.

6.2.2. *Based on the Discrete Logarithm*

Let p be a large prime, let α generate $\mathbb{Z}_p^*$ and let u be the smallest integer such that 2^u does not divide $p-1$. Given any $s \in \mathbb{Z}_p^*$, it is easy to compute the $u-1$ least significant bits of the unique e such that $0 \leqslant e \leqslant p-2$ and $s \equiv \alpha^e \pmod{p}$. Under the intractability assumption of the discrete logarithm, however, it is infeasible to learn anything about the uth least significant bit of e, because this problem is as hard as that of the discrete logarithm itself [Pe].

At the outset of the protocol, Peggy and Vic agree on p and α exactly as in Section 6.1.2. Let u be as above. The blobs are now defined by the sets $X = \mathbb{Z}_p^*$, $Y = \{0, 1, 2, ..., p-2\}$, and

$$v(x, y) = \begin{cases} y_u & \text{if } x \equiv \alpha^y \pmod{p} \\ \bullet & \text{otherwise,} \end{cases}$$

where y_u denotes the uth least-significant bit of y.

Property (i) holds because whenever she wishes to commit to some bit b, Peggy randomly chooses a $y \in \mathbb{Z}_p^*$ such that $y_u = b$ and computes $x = \alpha^y \bmod p$. She gives x to Vic but keeps y secret as her witness that allows her to open blob x as bit b. (Contrary to Section 6.2.1, Peggy must remember y in order to open x because she could not recompute it from x.) Property (ii) holds unconditionally, because α is a generator of $\mathbb{Z}_p^*$; hence the discrete logarithm of x is uniquely defined. Property (iii) holds from a computational point of view under the strengthened discrete-logarithm assumption (see Section 6.1.2).

6.2.3. *Based on any Probabilistic Encryption Schemes* [GMW]

A probabilistic encryption scheme in the sense of [GM] is a polynomial-time computable function $f\colon B \times Y \to X$ that, on input $b \in B$ and "coin tosses" $y \in Y$, outputs an encryption $f(b, y)$ of b. Decryption is unique: $f(b_1, y_1) = f(b_2, y_2)$ implies that $b_1 = b_2$. However, it is assumed to be computationally infeasible to learn anything about b from $f(b, y)$ without knowledge of some "trap-door" information.

Taking $B = \{0, 1\}$, we define blobs by

$$v(x, y) = \begin{cases} 0 & \text{if } f(0, y) = x \\ 1 & \text{if } f(1, y) = x \\ \bullet & \text{otherwise.} \end{cases}$$

The reader can easily verify that all the defining blob properties hold.

6.3. *Blobs That No Amount of Computing Power Can Break*

6.3.1. *Quantum Blobs* [BB3]

We assume in this section that the reader is familiar with the principles of quantum cryptography [BB1, BB2]. Charles H. Bennett has suggested that blobs could be implemented with similar principles. Indeed, quantum blobs can be implemented by a process very similar to quantum coin-tossing [BB2], which we do not repeat here. Let us say only that it can be proven that any cheating successfully conducted by Vic would lead to an apparatus capable of transmitting information faster than the speed of light. In principle, the Einstein–Rosen–Podolsky "paradox" [EPR, M] allows Peggy to cheat, exactly as with the coin-tossing protocol, but the technology needed to perform this cheating is far beyond any in the foreseeable future. More details are forthcoming [BB3].

6.3.2. *Multi-party Blobs* [CCD]

Blobs unconditionally secure for all parties can be obtained in a multiparty environment, assuming that more than two thirds of the participants are honest—in some cases only one half suffices—and that each pair of participants shares a private channel. Even a coalition of nearly a third of the participants with unlimited computing power cannot cheat the honest ones. For more details, consult [CCD].

6.4. *Oblivious Transfer, ANDOS, and Blobs*

Oblivious transfer is a powerful tool invented by Rabin [R4]. It involves two parties: Sam (the sender) and Rachel (the receiver). In the protocol's simplest form, Sam transmits one bit to Rachel in such a way that she has a 50% chance of receiving it. Neither party can influence whether or not the transmission will be successful. At the end of the transmission, Rachel knows whether she received the transmitted bit, but Sam does not know unless Rachel tells him.

ANDOS stands for "all-or-nothing disclosure of secrets." It is a tool invented by Wiesner [W] and investigated further by Brassard, Crépeau, and Robert [BCR1, BCR2] and Chaum [Ch3]. Here, Sam owns n secret bit strings $s_1, s_2, ..., s_n$. The ANDOS protocol allows Rachel to choose any i, $1 \leqslant i \leqslant n$, and obtain s_i from Sam in such a way that he cannot tell which secret she got. On the other hand, the protocol does not allow Rachel to learn anything about more than one of Sam's secrets.

The connection between ANDOS and the seemingly simpler oblivious transfer is achieved by a sequence of reductions discussed in [BCR1, Cr]: any protocol for the oblivious transfer can be transformed efficiently into a protocol for ANDOS. Moreover, if the underlying oblivious transfer protocol is unconditionally secure for Rachel (that is, if there is no way Sam can learn anything without Rachel's help about whether the transmitted bit was received), then the corresponding ANDOS protocol is also unconditionally secure for Rachel (that is, there is no way Sam can learn anything without Rachel's help about which secret she requested).

Blobs can be obtained easily from ANDOS and thus from oblivious transfer. For this purpose, Peggy assumes the role of Rachel, and Vic that of Sam. When she wants to commit to some bit b, Peggy asks Vic to prepare two secret random bit strings s_0 and s_1, and she obtains s_b through ANDOS. At this point, Vic has no idea whether Peggy knows s_0 or s_1; but she cannot know both of them. In order to open the blob, it suffices for Peggy to show Vic whichever string she had requested. She has an exponentially small chance of guessing the other string if she tries to cheat in opening the blob. Such attempted violation of property (ii), however, poses no appreciably greater risk to Vic than the exponentially small chance that Peggy already has (in the protocol of Section 2) of correctly guessing every challenge Vic will issue.

Blobs can also be obtained directly from oblivious transfer. For this, let Peggy assume the role of Sam, and Vic that of Rachel. In order to commit to some bit b, Peggy chooses a Boolean matrix M at random, except that the parity of the number of 1's in each row is equal to b. Peggy then sends Vic, in some agreed order, each bit of M separately by means of oblivious transfer. At this point, nothing is revealed about b unless Vic was lucky enough to obtain each and every bit in at least one row of M, which is exponentially unlikely in the number of columns. In order to open the blob, Peggy gives Vic all the bits of M in the clear. Peggy can attempt to cheat by lying about one bit in each row, in the hope that Vic obtained none of these bits during the oblivious transfer phase; but this is exponentially unlikely in the number of rows.

The blobs based on ANDOS and those based directly on oblivious transfer are dual of each other, just as the blobs described in Section 6.1 are dual of the blobs described in Section 6.2. Assume, for instance, that the underlying oblivious transfer protocol is unconditionally secure for the receiver. The blobs based on ANDOS are then unconditionally secure for Peggy (as in Section 6.1): nothing Vic can do will enable him to learn anything about the bit committed to by Peggy unless she opens it. On the other hand, the blobs based directly on oblivious transfer are unconditionally secure for Vic (as in Section 6.2): although Peggy could successfully cheat in opening a blob, she can only do this by being extremely lucky—no amount of computing power can help her. Both implementations are unconditionally secure for Peggy and for Vic (except with exponentially small probability) if the underlying oblivious transfer protocol is unconditionally secure for both parties.

Even though oblivious transfer is conceptually simpler than bit commitment, it seems to be an inherently more powerful primitive. Indeed, we do not know of any construction capable of achieving oblivious transfer from blobs that satisfy only defining properties (i) through (iv) (although this becomes possible if we add an appropriate "trap-door" property to the definition of blobs). Moreover, bit commitment is a universal primitive for minimum disclosure, whereas oblivious transfer is a universal primitive for the more general multiparty computation in which all parties have secrets [CDG].

6.5. *Proving Blob Equality and Inequality*

In this section, we address the question of how Peggy can convince Vic that she can open any two blobs she has committed to, and show the same bit or opposite bits, whichever is the case, without disclosing anything else. Referring to Section 5, we distinguish between property (v), which allows Peggy to convince Vic that she can open two blobs to show the same bit (provided this is the case), and property (vi), which allows Peggy to convince Vic that she can open two blobs to show different bits (provided this is the case).

Several specific implementations of blobs given above (Sections 6.1.1, 6.1.2, 6.1.3, and 6.2.1) allow property (v) to be obtained as a primitive operation. Consider the blobs of Section 6.1.1 or 6.2.1, for instance. If $x \equiv y^2 s^b (\bmod n)$ and $\hat{x} \equiv \hat{y}^2 s^b (\bmod n)$ for the same bit b, then if Peggy computes $z = y\hat{y}s^b \bmod n$ and gives it to Vic, will be convinced that she can open x and $\hat{x}$ to show the same bit after checking that $x\hat{x} \equiv z^2 \pmod n$. However, if we use a general probabilistic encryption scheme to implement blobs (Section 3.2.3), it is not obvious that property (v) is always so easy to obtain. We challenge the reader to figure out how blob equality can be achieved with oblivious transfer blobs.

Although property (vi) is also easy to implement as a primitive operation with the blobs of Sections 6.1.1, 6.1.2, and 6.2.1, it is intriguing to note that this does not seem to be so with the blobs based on graph isomorphism (Section 6.1.3). This shows that there is a fundamental difference between properties (v) and (vi): it is easy for Peggy to convince Vic that she knows an isomorphism between two graphs when this is so, but how could she convince him that she does *not* know such an isomorphism? (Notice that this question has nothing to do with whether or not graph non-isomorphism is in NP.)

Even though they may not always be achieved as primitive operations, it turns out that properties (v) and (vi) can always be obtained through an interactive sub-protocol. For simplicity, let us assume for the moment that our blobs are described by sets of integers X and Y, and by a verification function v, as in Sections 6.1 and 6.2 (we thus temporarily rule out oblivious transfer blobs, quantum blobs, and multi-party blobs). Ivan Damgård has pointed out that the basic protocol of Section 2 can be used for this purpose. Assume, for instance, that Peggy can open blobs x and $\hat{x}$ to show the same bit and that she would give Vic y and $\hat{y}$ if she wanted to open these blobs. Instead of showing y and $\hat{y}$, she uses the basic protocol to convince Vic that she knows y and $\hat{y}$ such that $v(x, y) = v(\hat{x}, \hat{y}) \in \{0, 1\}$, which is an NP statement!

It is interesting to note that *any* bit-commitment scheme can be transformed into one that also has properties (v) and (vi): it is enough to assume the abstract defining blob properties (i) through (iv). This can be achieved by an extension of an idea first suggested by Charles H. Bennett. The construction will be described in the forthcoming paper on quantum cryptography [BB3].

For blobs that already offer property (v) as a primitive, but not property (vi) (such as the blobs based on graph isomorphism of Section 6.1.3), it is more efficient to implement property (vi) through a subprotocol that makes use of property (v). Let x and $\hat{x}$ be two blobs that Peggy can open to show distinct bits. To convince Vic of this fact, Peggy commits to two more blobs z and $\hat{z}$, claiming that she can open them to show distinct bits. At this point, Vic issues one of two possible challenges. As a result, Peggy must either open both z and $\hat{z}$, thus showing a 0 and a 1, or she must use property (v) twice to convince Vic of the equivalence between x and z (or $\hat{z}$, whichever is the case) and the equivalence between $\hat{x}$ and $\hat{z}$ (or z). If the above is repeated k times and, in fact, she could only open x and $\hat{x}$ to show the same bit, Peggy has only a probability 2^{-k} of successful cheating.

The reverse process is obvious: if Peggy knows how to convince Vic that she can open two given blobs to show distinct bits when this is so, and if she wishes to convince him that she can open blobs x and $\hat{x}$ to show the same bit, she simply commits to an appropriate blob x'; then she convinces him, using property (vi), that she could open x and x' to show distinct bits and that she could also open $\hat{x}$ and x' to show distinct bits.

6.6. *Multivalued Blobs*

Consider any finite set D. The notion of bit commitment generalizes naturally to that of commitment to an arbitrary member of D. If D contains k elements, the effect of such a commitment could obviously be obtained through commitments to $\log_2 k$ ordinary blobs. However, it would be more interesting from an efficiency point of view if this could be achieved by committing to a single "multivalued blob."

In the context of our basic protocol from Section 2, truth table entries are never opened in isolation. Whenever Peggy opens one such entry, she always opens the entire row containing it. We can therefore speed up the entire basic protocol nearly threefold if each blob can be opened in 8 different ways—corresponding to each possible choice of three bits in a truth table row. Of course, this idea is interesting only if the commitment to and opening of multivalued blobs are not substantially more expensive than the corresponding operations on binary blobs.

Some of our previously discussed implementations extend easily to multivalued blobs. This is the case for both implementations based on the discrete logarithm (Sections 6.1.2 and 6.2.2). In order to generalize the blobs of Section 6.1.2, Vic gives Peggy k distinct values $s_1, s_2, ..., s_k$ in $\mathbb{Z}_p^*$ at the outset of the protocol (each $s_i \neq 1$). In order to commit to the ith element of D, Peggy chooses a random integer y, $0 \leqslant y \leqslant p-2$ and computes the blob $x = s_i \alpha^y \bmod p$. The protocol of [CEG] is in order if it is desirable that Vic convince Peggy that he knows the logarithms of all these s_i's—either to obtain chameleon blobs or to remove the requirement that the factoring of $p-1$ be known (as discussed at the end of Section 6.1.2).

The blobs of Section 6.2.2 generalize easily, because the $\log \log p$ successive higher-order bits after the uth least significant bit of a discrete logarithm are simultaneously as secure as the uth least significant bit alone [Pe]. Using these bits, $(\log p)$-valued blobs can be realized.

7. Is it Better to Trust the Prover or the Verifier?

"Cheating" takes on different meanings, according to whether one is talking about Peggy or about Vic. For Vic to cheat means that he learns something beyond the fact that Peggy has access to the information she claims to have. Perhaps he did not quite obtain the Hamiltonian circuit he is desperately seeking, for instance, but he learned enough to drastically reduce his search. On the other hand, for Peggy to cheat means that she succeeds in convincing Vic that she has information that would pass the certifying procedure, when in fact she does not.

It is also interesting to distinguish between *lucky* and *daring* successful cheating. The former refers to Peggy or Vic figuring out—against all odds—a piece of information that will enable him/her to quietly go about his/her cheating with the certainty of being successful and undetected. The latter refers to Peggy or Vic making an illegal move that is almost certainly going to result in his/her cheating being detected at some point in the future, but that might nonetheless, with an exponentially small probability, allow him/her to succeed. Finally, cheating may be called *retroactive* (or *off-line*) if it can take place some time after the protocol is completed, by looking back at its transcript; it is *real-time* (or *on-line*) if it must be completed while the protocol is taking place.

If the blobs of Section 6.2 are used (corresponding to the protocols previously given by [GMW, BC1]), Peggy could never participate with peace of mind: an algorithmic breakthrough might allow Vic to cheat retroactively, even if the new algorithm is not fast enough for a real-time response. Even if the cryptographic assumption turns out to be well founded, Vic still has a (very slight) probability of lucky (hence undetectable) cheating. On the other hand, regardless of any assumptions, the only cheating Peggy could attempt would be of the daring kind.

By contrast, if the blobs of Section 6.1.1 are used (corresponding to the protocols previously given in [Ch4, BC2]), the *only* way Vic can hope to learn anything about Peggy's secret is to be daring right from the beginning and to choose a quadratic non-residue as his s. He would almost certainly get caught by Peggy while trying to convince her that s is a quadratic residue, but would, if successful, be capable of distinguishing blobs used by Peggy as commitments to 0 from those used for 1. Asking Vic to disclose a square root of s at the end of the protocol (which is not detrimental to him at that point, assuming he is honest) provides Peggy with *certainty* that Vic has not learned any of her secrets (and *never* will retroactively). If the blobs of Section 6.1.2 are used, even this unlikely opportunity for daring cheating is not available to Vic. On the other hand, with the implementations of Section 6.1, Vic's belief that Peggy cannot cheat depends on his belief in the appropriate cryptographic assumption. With the implementation of Section 6.1.1, for instance, Peggy could "open" any quadratic residue as either 0 or 1, whichever suits her best, if she could only obtain a square root of s before the end of the first round in which she is asked a challenge she is not otherwise prepared to answer. (Obtaining this square root at any later time would be of no use to her.) Moreover, even if the cryptographic assumption is well founded, Peggy still has a (very slight) possibility of breaking it by luck, but she must be daring to suggest conducting the protocol in the hope that she will be so lucky. Finally, retroactive cheating is meaningless for either party in this context. An algorithm capable of factoring in two weeks, for instance, would spell doom to the protocol if blobs were implemented as in Section 6.2.1, but it would be of no immediate consequence with the blobs of Section 6.1.1.

If blobs unconditionally secure for Peggy are used (Section 6.1), additional security for Vic is obtained by asking Peggy to repeat the entire protocol with a different type of blob each time (as pointed out originally in [Ch4]). In order to cheat, Peggy would then have to be capable of breaking several different cryptographic assumptions. For instance, she would need efficient on-line algorithms both for factoring and for extracting discrete logarithms. Curiously, the opposite effect is obtained with the blobs that are unconditionally secure for Vic (Section 6.2): repeating the protocol with different types of blobs would only make it *easier* for Vic to cheat, since he can do so by breaking (possibly off-line) any one of the underlying cryptographic assumptions. Nonetheless, increased security *can* be obtained if several types of blobs unconditionally secure for Vic are combined in a different way: each time Peggy wishes to commit to some bit b, she commits to one blob of each type at random except that b is the exclusive-or of the corresponding bits. Naturally, using this strategy with blobs unconditionally secure for Peggy would only make it easier for her to cheat.

Is it preferable to trust Vic or Peggy? We do not know, but it sure is nice to have the choice! Finally, consider the following provocative situation: suppose that Peggy claims to have proven Theorem **T** and she uses the blobs of Section 6.1.1 to convince a skeptical Vic of this. At the end of the protocol, regardless of any unproved assumptions, Vic will be convinced that Peggy has either a proof of **T** or hot results on integer factoring! In particular, no assumptions are needed if **T**'s claim is: "I have an efficient factoring algorithm...."

ACKNOWLEDGMENTS

We wish to thank Josh Benaloh, Charles H. Bennett, Joan Boyar, Ivan Damgård, Jan-Hendrik Evertse, Joan Feigenbaum, Lance Fortnow, Shafi Goldwasser, Oded Goldreich, Johan Hastad, Russel Impagliazzo, Leonid Levin, Silvio Micali, Bill Neven, Jean-Marc Robert, Steven Rudich, Adi Shamir, Jeroen van de Graaf, and Moti Yung for fruitful discussions.

REFERENCES

[AH] L. M. ADLEMAN AND M.-D. A. HUANG, Recognizing primes in random polynomial time, *in* "Proceedings, 19th Annual ACM Symposium on the Theory of Computing, May 1987," pp. 462–469.

[Ba] L. BABAI, Trading group theory for randomness, *in* "Proceedings, 17th Annual ACM Symposium on the Theory of Computing, May 1985," pp. 421–429.

[Be] J. D. BENALOH (COHEN), Cryptographic capsules: A disjunctive primitive for interactive protocols, *in* "Advances in Cryptology: Proceedings, CRYPTO '86, Santa Barbara, CA, August 1986," pp. 213–222, Springer-Verlag, New York/Berlin, 1987.

[BBDGQ] S. Bengio, G. Brassard, Y. Desmedt, C. Goutier, and J.-J. Quisquater, Aspects and importance of secure implementations of identification systems, Manuscript M209, Philips Research Laboratory, Brussels, Belgium, 1987.

[BB1] C. H. Bennett and G. Brassard, An update on quantum cryptography, *in* "Advances in Cryptology: Proceedings, CRYPTO '84, Santa Barbara, CA, August 1984," pp. 474–480, Springer-Verlag, New York/Berlin, 1985.

[BB2] C. H. Bennett and G. Brassard, Quantum cryptography: Public-key distribution and coin-tossing, *in* "IEEE International Conference on Computers, Systems and Signal Processing, Bangalore, India, December 1984," pp. 175–179.

[BB3] C. H. Bennett and G. Brassard, Quantum cryptography, in preparation.

[Bl] M. Blum, Coin flipping by telephone: A protocol for solving impossible problems, "Proceedings of the 24th IEEE Compcon, 1982," pp. 133–137; reprinted in *SIGACT News* **15**, (1983), 23–27.

[BKK] J. F. Boyar, M. W. Krentel, and S. A. Kurtz, "A Discrete Logarithm Implementation of Zero-Knowledge Blobs," Technical Report 87-002, University of Chicago, March 1987.

[BB] G. Brassard and P. Bratley, "Algorithmics: Theory and Practice," Prentice-Hall, Englewood Cliffs, NJ, 1988.

[BC1] G. Brassard and C. Crépeau, Zero-knowledge simulation of Boolean circuits, *in* "Advances in Cryptology: Proceedings, CRYPTO '86, Santa Barbara, CA, August 1986," pp. 223–233, Springer-Verlag, New York/Berlin, 1987.

[BC2] G. Brassard and C. Crépeau, Non-transitive transfer of confidence: A *perfect* zero-knowledge interactive protocol for SAT and beyond, *in* "Proceedings, 27th Annual IEEE Symposium on the Foundations of Computer Science, October 1986," pp. 188–195.

[BCR1] G. Brassard, C. Crépeau, and J.-M. Robert, Information theoretic reductions among disclosure problems, *in* "Proceedings, 27th Annual IEEE Symposium on the Foundations of Computer Science, October 1986," pp. 168–173.

[BCR2] G. Brassard, C. Crépeau, and J.-M. Robert, All-or-nothing disclosure of secrets, *in* "Advances in Cryptology: Proceedings, CRYPTO '86, Santa Barbara, CA, August 1986," pp. 234–238, Springer-Verlag, New York/Berlin, 1987.

[Ch1] D. Chaum, Showing credentials without identification: Signatures transferred between unconditionally unlinkable pseudonyms, presented at "EUROCRYPT '85, Linz, Austria, April 1985."

[Ch2] D. Chaum, Security without identification: Transaction systems to make Big Brother obsolete, *Comm. ACM* **28**, No. 10 (1985), 1030–1044.

[Ch3] D. Chaum, Showing satisfiability without revealing how, presented at "Conference on Algorithms, Randomness and Complexity, Marseille/Lumigny, March 1986."

[Ch4] D. Chaum, Demonstrating that a public predicate can be satisfied without revealing any information about how, *in* "Advances in Cryptology: Proceedings, CRYPTO '86, Santa Barbara, CA, August 1986," pp. 195–199, Springer-Verlag, New York/Berlin, 1987.

[CCD] D. Chaum, C. Crépeau, and I. B. Damgård, Multiparty unconditionally secure protocols, *in* "Proceedings, 20th Annual ACM Symposium on the Theory of Computing, May 1988," pp. 11–19.

[CDG] D. Chaum, I. B. Damgård, and J. van de Graaf, Multiparty computations ensuring privacy of each party's input and correctness of the result, *in* "Advances in Cryptology: Proceedings, CRYPTO '87, Santa Barbara, CA, August 1987," pp. 87–119, Springer-Verlag, New York/Berlin, 1988.

[CEG] D. Chaum, J.-H. Evertse, and J. van de Graaf, An improved protocol for demonstrating possession of a discrete logarithm and some generalizations, *in* "Advances in Cryptology: Proceedings, EUROCRYPT '87, Amsterdam, The Netherlands, April 1987," pp. 127–141, Springer-Verlag, New York/Berlin, 1988.

[CEGP] D. Chaum, J.-H. Evertse, J. van de Graaf, and R. Peralta, Demonstrating prossession of a discrete logarithm without revealing it, *in* "Advances in Cryptology: Proceedings, CRYPTO '86, Santa Barbara, CA, August 1986," pp. 200–212, Springer-Verlag, Berlin/New York, 1987.

[Co] S. A. Cook, The complexity of theorem proving procedures, *in* "Proceedings, 3rd Annual ACM Symposium on the Theory of Computing, 1971," pp. 151–158.

[Cr] C. Crépeau, Equivalence between two flavours of oblivious transfers, *in* "Advances in Cryptology: Proceedings, CRYPTO '87, Santa Barbara, CA, August 1987," pp. 350–354, Springer-Verlag, New York/Berlin, 1988.

[DH] W. Diffie and M. E. Hellman, New directions in cryptography, *IEEE Trans. Inform. Theory* **IT-22** (1976), 644–654.

[EPR] A. Einstein, B. Podolsky, and N. Rosen, *Phys. Rev.* **47** (1935), 777.

[FFS] U. Feige, A. Fiat, and A. Shamir, Zero knowledge proofs of identity, *in* "Proceedings, 19th Annual ACM Symposium on the Theory of Computing, May 1987," pp. 210–217.

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

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

[GJ] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," Freeman, New York, 1979.

[G] J. Gill, Computational complexity of probabilistic Turing machines, *SIAM J. Comput.* **6**, No. 4 (1977), 675–695.

[GMW] O. Goldreich, S. Micali, and A. Wigderson, Proofs that yield nothing but their validity and a methodology of cryptographic protocol design, *in* "Proceedings, 27th Annual IEEE Symposium on the Foundations of Computer Science, October 1986," pp. 174–187; originally presented at "Conference on Algorithms, Randomness and Complexity, Marseille/Lumigny, March 1986."

[GK] S. Goldwasser and J. Kilian, Almost all primes can be quickly certified, *in* "Proceedings, 18th Annual ACM Symposium on the Theory of Computing, May 1986," pp. 316–329.

[GM] S. Goldwasser and S. Micali, Probabilistic encryption, *J. Comput. System Sci.* **28**, No. 2 (1984), 270–299.

[GMR] S. Goldwasser, S. Micali, and C. Rackoff, The knowledge complexity of interactive proof-systems, *in* "Proceedings, 17th Annual ACM Symposium on the Theory of Computing, May 1985," pp. 291–304.

[GS] S. Goldwasser and M. Sipser, Arthur–Merlin games versus interactive proof systems, *in* "Proceedings, 18th Annual ACM Symposium on the Theory of Computing, May 1986," pp. 59–68.

[M] N. D. Mermin, Bringing home the atomic world: Quantum mysteries for anybody, *Amer. J. Phys.* **49**, No. 10 (1981), 940–943.

[Pa] C. H. Papadimitriou, Games against nature, *J. Comput. System Sci.* **31** (1985), 288–301.

[Pe] R. Peralta, Simultaneous security of bits in the discrete log, *in* "Advances in Cryptology: Proceedings, EUROCRYPT '85, Linz, Austria, April 1985," pp. 62–72, Springer-Verlag, New York/Berlin, 1986.

[PG] R. Peralta and J. van de Graaf, A simple way to show the validity of your public key, *in* "Advances in Cryptology: Proceedings, CRYPTO '87, Santa Barbara, CA, August 1987," pp. 128–134, Springer-Verlag, New York/Berlin, 1988.

[Pr] V. Pratt, Every prime has a succinct certificate, *SIAM J. Comput.* **4** (1975), 214–220.

[R1] M. O. Rabin, Probabilistic algorithms, *in* "Algorithms and Their Complexity: Recent Results and New Directions" (J. F. Traub, Ed.), pp. 21–39, Academic Press, New York, 1976.

[R2] M. O. Rabin, Digitalized signatures, *in* "Foundations of Secure Computation" (R. A. DeMillo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, Eds.), pp. 155–168, Academic Press, New York, 1978.

[R3] M. O. RABIN, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization," MIT/LCS/TR-22, 1979.

[R4] M. O. RABIN, "How to Exchange Secrets by Oblivious Transfer," Technical Memo TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[S] C. E. SHANNON, A mathematical theory of communication, *Bell System Tech. J.* **27** (1948), 379–423, 623–656.

[SS] R. SOLOVAY AND V. STRASSEN, A fast Monte Carlo test for primality, *SIAM J. Comput.* **6** (1977), 84–85.

[W] S. WIESNER, Conjugate coding, unpublished manuscript *ca* 1970; *SIGACT News* **15** (1983), 78–88.