

# On Monotone Formula Closure of SZK

Alfredo De Santis <sup>\*</sup>    Giovanni Di Crescenzo <sup>†</sup>    Giuseppe Persiano <sup>‡</sup>  
Moti Yung <sup>§</sup>

## Abstract

We investigate structural properties of statistical zero knowledge (SZK) both in the interactive and in the non-interactive model. Specifically, we look into the closure properties of SZK languages under monotone logical formula composition. This gives rise to new protocol techniques.

We show that interactive SZK for random self reducible languages (RSR) (and for co-RSR) is closed under monotone boolean operations. Namely, we give SZK proofs for monotone boolean formulae whose atoms are statements about an SZK language which is RSR (or a complement of RSR). All previously known languages in SZK are in these classes.

We then show that if a language  $L$  has a non-interactive SZK proof system then honest-verifier interactive SZK proof systems exist for all monotone boolean formulae whose atoms are statements about the complement of  $L$ .

We also discuss extensions and generalizations.

## 1 Introduction

Goldwasser, Micali, and Rackoff [34] introduced the notion of a *zero-knowledge proof*, a proof procedure with the remarkable property of yielding nothing but the validity of the assertion. Goldreich, Micali and Wigderson [30] then showed that under the assumption of the existence of one-way functions all

NP statements have zero-knowledge proofs (thus introducing computational zero-knowledge).

Statistical zero-knowledge (SZK) proofs [34] are those proofs that can be shown zero knowledge without resorting to any unproven computational assumption, and thus are as secure as the input statement (language) itself. The notion is important, both from a practical prospective and a theoretical one.

From a practical point of view, note that typical SZK proofs of number theoretic statements are the most efficient zero-knowledge proofs. Thus, naturally they are the ones employed in practical and working systems, e.g., they are employed in identification procedures, as was first noted in [23] (and was followed by many practical schemes). Identification schemes rely on the fact that certain SZK proofs are also “proofs of possession of knowledge of a witness” [23, 43, 4] (a notion alluded to in [34, 25]). Number theoretic SZK proofs are also the basic schemes used in initializing further zero-knowledge proof systems (as is done in the non-interactive scenario and its extensive applications, e.g. [12, 13]). Thus, from the protocol design prospective, a richer set of SZK protocol techniques for more involved yet well-formed statements in SZK may find further applications.

The following “multi faceted identification scheme” is an example of an application of an extended set of SZK languages. Consider a group of provers each knowing a witness to a different disjunct of a disjunctive boolean formula. The provers share an identification scheme in the following sense. At certain times a prover can identify itself as being a member of the group (by proving validity of the entire formula, without disclosing its identity which may relate to the specific disjunct she/he knows a witness of). On the other hand, at other times the same prover can verify its individual identity (by proving a possession of a witness to the specific disjunct associated with her/him). Note that each disjunct can, in fact, be a very efficient quadratic residue based statement. Other similar

---

<sup>\*</sup>Dipartimento di Informatica ed Appl., Università di Salerno, 84081 Baronissi (SA), Italy. e-mail: ads@udsab.dia.unisa.it. Part of this work was done while visiting the International Computer Science Institute, Berkeley, CA.

<sup>†</sup>Computer Science Department, University of California at San Diego and Dipartimento di Informatica ed Appl., Università di Salerno, 84081 Baronissi (SA), Italy.

<sup>‡</sup>Dipartimento di Matematica, Università di Catania, 95125 Catania, Italy. e-mail: giuper@udsab.dia.unisa.it. Part of this work was done while visiting DIMACS Center, Piscataway, NJ.

<sup>§</sup>IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY. e-mail: moti@watson.ibm.com

applications may employ a more complicated sharing of witnesses according to some given structure (typically called an access structure [36]).

We remark that SZK proofs also motivated the useful notion of computationally sound proofs (in conjunction with the idea of zero-knowledge arguments) [17, 18, 38], and inspired the recent work on program checking and testing [14, 11].

From theoretical prospective, SZK (and the related more stringent notion of perfect ZK (PZK)) have been the subject of a number of early investigations. Goldreich, Micali and Wigderson [30] gave perfect zero-knowledge proofs for the language of graph isomorphism as well as of graph nonisomorphism (which was the first language with interactive proof not known to be in NP). It has been proved that a complement of any SZK language has a 2-round proof system [26, 2], and that any language in SZK also has a 2-round proof system [2]. The round complexity and prover-power for SZK were studied in [5, 6]. As statistical zero-knowledge proofs do not depend on the properties of assumed hard problems (like one-way functions) but only on the input language, they constitute the clean context for studying the intrinsic structural properties of the notion of a zero-knowledge proof and knowledge-complexity of languages [34, 33, 32]; this is also part of our motivation as we review below.

Given its extensive practical use, its influence, and the complexity theoretic investigations concerning SZK, we observed that relatively very few languages are known to be in SZK. This, in turn, implies that only very few general protocol techniques are available for designing and utilizing the notion. Besides the proofs for quadratic residuosity and quadratic non residuosity given in the original paper [34], statistical zero-knowledge proofs were given in [30, 27, 43, 29, 15], and in [12, 20] for the non-interactive zero-knowledge case. We note that all these languages have certain relations to random self-reducibility properties – either positively: claiming that the input is randomly reducible (by a group-action) to some structure (e.g., one graph isomorphic to another), or negatively: claiming the input not to be reducible (e.g., a graph is not isomorphic to another) which we call a complement of RSR language; see e.g. [43]. All of their proof systems rely on certain algebraic properties associated with the random self reducibility property.

Another motivation to our work has been the work of Goldreich and Petrank [33]. They build on the initial suggestion of knowledge-complexity

given in [34] and formally define the concept of a proof that leaks  $k$  bits of knowledge (with zero-knowledge being the case  $k = 0$ ). Note that for computational zero-knowledge (assuming one-way functions or secure envelopes), it is known that everything that has an interactive proof (namely, PSPACE [42, 37]), has a zero-knowledge proof [35, 10]. The question is not as clear for statistical zero-knowledge. That is, are there languages in Statistical- $KC(1)$  (that is the class of languages that can be proved releasing exactly 1 bit of knowledge) but not in Statistical- $KC(0)$ ? Or in general, in Statistical- $KC(k)$  but not in Statistical- $KC(k - 1)$ ? From [26, 2], combined with [16], we know that, unless the polynomial hierarchy collapses, NP-complete languages are not likely to be in Statistical- $KC(0)$ . Also, recent results of Goldreich, Ostrovsky and Petrank [32] (building on [7]) seems to imply that, PSPACE is at least in statistical- $KC(\log n)$  (as a smaller bit knowledge complexity for PSPACE would imply that the polynomial hierarchy collapses).

From the point of view of understanding the power and possibility of statistical- $KC(0)=SZK$  and designing protocol techniques, the most natural candidates for SZK are languages that can be constructed from languages in SZK using general combining mechanisms (i.e., it is known that sequential composition of languages maintains appropriately defined zero-knowledge-ness (see [31, 28])). The results of this paper suggest new protocol techniques and prove that a large class of logically constructible languages over atomic languages which are in SZK are themselves in SZK.

## Summary of results:

**The interactive model.** We prove that interactive perfect zero-knowledge for random self reducible languages (see [43]) is closed under monotone formulae. More precisely, we present perfect ZK proofs for all *monotone boolean formulae* whose atoms are statements about membership in a random self reducible language. This is extended to atoms over a set of languages: either statements over random self reducible languages (like isomorphism, residuosity), or ones where the complement language is random self reducible (like non-isomorphism, non-residuosity). Moreover, we prove that *threshold formulae* also have perfect zero-knowledge proofs.

**The non-interactive model.** We study the properties of statistical zero knowledge in the non-interactive shared-string model (NISZK) of [13, 12]. We show that if a language  $L$  has a non-interactive SZK proof system then honest-verifier interactive SZK proof systems exist for all monotone boolean formulae whose atoms are statements about the complement of  $L$ . (Note that honest verifier SZK proof can be turned into an SZK proof assuming one-way permutations).

**Other results.** We also show monotone formulae closure of proof systems for some specific class of *non-monotone formulae*. Finally, we pose new open questions and directions regarding closure properties of SZK, and protocol techniques.

Due to page limitation, formal proofs of our results will appear in the final version of the paper.

## 2 Notation and terminology

A monotone formula  $\phi$  over the variables  $v_1, \dots, v_m$  is a tree where each internal node is either an OR gate or an AND gate with 2 input edges and one output edge and each leaf is labeled with a variable. Each truth assignment  $t : \{v_1, \dots, v_m\} \rightarrow \{0, 1\}$  defines naturally the validity  $t(\phi)$  of the formula under  $t$ .

Let  $L$  be a language and let  $\phi$  be a monotone formula over the literals  $v_1, \dots, v_m$ . Let  $\vec{x} = (x_1, \dots, x_m)$  be an  $m$ -tuple of binary strings and let  $\chi_L$  denote the indicator function for the language  $L$  (i.e.,  $\chi_L(x) = 1$  iff  $x \in L$ ). Then we say that  $(\phi, \vec{x}) \in \Phi(L)$  if and only if  $\phi(\chi_L(x_1), \dots, \chi_L(x_m)) = 1$ . Let us give a concrete example. Consider the language GI of pairs of isomorphic graphs, let  $\phi$  be the formula  $(v_1 \wedge v_2) \vee (v_3 \wedge v_4)$  and let  $\vec{G} = (G_{10}, G_{11}, G_{20}, \dots, G_{41})$  be an 8-tuple of graphs. Then, the pair  $(\phi, \vec{G})$  belongs to the language  $\Phi(\text{GI})$  iff the following statement is true

$$((G_{10} \approx G_{11}) \wedge (G_{20} \approx G_{21})) \vee ((G_{30} \approx G_{31}) \wedge (G_{40} \approx G_{41})).$$

**Read-once monotone formulae.** Throughout this paper we will only consider read-once monotone formulae. This is, however, without loss of generality, as we can always have a read-once formula by a small change in the representation (and length extension) which is still in the language. Suppose that  $\phi$  is a formula on  $m$  variables  $v_1, \dots, v_m$  and that  $v_i$  appears  $l_i$  times. By giving

each variable a different name each time it appears, we construct a formula  $\phi'$  on  $m' = \sum_{i=1}^m l_i$  variables such that

$$\phi(v_1, \dots, v_m) = \phi'(\underbrace{v_1, \dots, v_1}_{l_1}, \dots, \underbrace{v_m, \dots, v_m}_{l_m}).$$

From this it follows that

$$(\phi, (x_1, \dots, x_m)) \in \Phi(L) \text{ if and only if}$$

$$(\phi', (\underbrace{x_1, \dots, x_1}_{l_1}, \dots, \underbrace{x_m, \dots, x_m}_{l_m})) \in \Phi(L).$$

## 3 Closure results in the interactive model

We start by reviewing the definition of perfect and statistical zero-knowledge of [34].

Let  $P$ , the prover, be a probabilistic Turing machine and  $V$ , the verifier, a probabilistic polynomial-time machine that share the same input and can communicate with each other. By  $P \leftrightarrow V(x)$  we denote  $V$ 's output after interacting with  $P$  on input  $x$ .

**Definition 1** *A pair  $(P, V)$  constitutes an interactive proof system for the language  $L$  if*

1. (Completeness) *If  $x \in L$  then*

$$\Pr(P \leftrightarrow V(x) = \text{ACCEPT}) \geq 2/3.$$

2. (Soundness) *If  $x \notin L$  then for all probabilistic Turing machines  $P^*$*

$$\Pr(P^* \leftrightarrow V(x) = \text{ACCEPT}) \leq 1/2.$$

We define  $View_V(x)$ ,  $V$ 's view of the interaction with  $P$  on input  $x$ , as the probability space that assigns to pairs  $(R; \log)$  the probability that  $R$  is the tape of  $V$ 's random bits and that  $\log$  is the transcript of a conversation between  $P$  and  $V$  on input  $x$  given that  $R$  is  $V$ 's random tape.

**Definition 2** *An interactive proof system  $(P, V)$  for  $L$  is a perfect zero-knowledge proof system for  $L$  (in short, a PZK proof system) if for each  $V^*$  there exists a probabilistic machine  $M_{V^*}$  (called the simulator) running in expected polynomial time such that for all  $x \in L$  the probability spaces  $View_{V^*}(x)$  and  $M_{V^*}(x)$  coincide.*

**Definition 3** *An interactive proof system  $(P, V)$  for  $L$  is a statistical zero-knowledge proof system*

for  $L$  (in short, a SZK proof system) if for each  $V^*$  there exists a probabilistic machine  $M_{V^*}$  (called the simulator) running in expected polynomial time such that for all constants  $c$  and all sufficiently long  $x \in L$

$$\sum_{\alpha} |\Pr(\text{View}_{V^*}(x) = \alpha) - \Pr(M_{V^*}(x) = \alpha)| < |x|^c.$$

We denote by PZK and SZK the classes of languages that have a PZK proof system and a SZK proof system, respectively. An important superclass of SZK is the class of *honest-verifier* SZK obtained by demanding that the simulator exist only for the designated verifier  $V$ .

To ease the exposition, we first present our results for the languages GI of isomorphic pairs of graphs and for GNI of non-isomorphic pairs of graphs. However, all our results apply to any random self reducible language (e.g., quadratic residuosity or generator (sub)-group) and for their complement. Moreover, the languages (corresponding to different literals of the formula) do not necessarily have to be related to the same random self reducible problem. For example, it is possible to modify our proof system to obtain perfect zero-knowledge proofs for propositions like “ $G$  is isomorphic to  $H$  OR  $y$  is a quadratic residue modulo  $x$ ”, mixing various RSR languages (same hold for co-RSR languages).

The rest of this section is divided into four parts. In the first part we give a perfect zero-knowledge proof system for the language  $\Phi(\text{GI})$ . Then, we present a perfect zero-knowledge proof system for the language  $\Phi(\text{GNI})$ . In the third part we deal with the case of a general random self reducible language. Finally, we discuss some extensions of our results.

### 3.1 A perfect zero-knowledge proof system for $\Phi(\text{GI})$

We now prove that the language  $\Phi(\text{GI})$  has a perfect zero-knowledge proof system.

**An informal discussion.** Let  $\phi$ , a monotone formula on  $m$  literals, and  $\vec{G} = (G_{10}, G_{11}, G_{20}, \dots, G_{m1})$  be the input to the proof system for  $\Phi(\text{GI})$ . Then the prover constructs and gives to the verifier an  $m$ -tuple  $\vec{H}$  of graphs with the following property. If  $(\phi, \vec{G}) \in \Phi(\text{GI})$ ,  $\vec{H}$  can be “opened” both as a 0 and as 1. On the other hand, if  $(\phi, \vec{G}) \notin \Phi(\text{GI})$  then each  $m$ -tuple  $\vec{H}$  can be opened

in at most one way. Then the verifier chooses a bit  $b$  at random and asks the prover to open  $\vec{H}$  as  $b$ . If the prover succeeds then the verifier accepts otherwise it rejects.

In order to be more precise, let us introduce a bit of notation. The *isomorphism value*  $C^{\text{iso}}(t, \phi)$  of a formula  $\phi$  under a truth assignment  $t$  is defined as follows. If the formula  $\phi$  consists of a single literal  $v$  then  $C^{\text{iso}}(t, \phi) = t(v)$ . Suppose that  $\phi = \phi_1 \wedge \phi_2$ . If the isomorphism values of  $\phi_1$  and  $\phi_2$  are equal, then the isomorphism value of  $\phi$  is equal to this value; otherwise we set  $C^{\text{iso}}(t, \phi)$  equal to undefined value  $-$ . Finally, suppose that  $\phi = \phi_1 \vee \phi_2$ . If the isomorphism values of  $\phi_1$  and  $\phi_2$  are both different from  $-$  then  $C^{\text{iso}}(t, \phi) = C^{\text{iso}}(t, \phi_1) \oplus C^{\text{iso}}(t, \phi_2)$ ; otherwise we set  $C^{\text{iso}}(t, \phi) = -$ . In the sequel we will denote by  $C^{\text{iso}}(c_1, \dots, c_m; \phi)$  the isomorphism-value of  $\phi$  under the assignment that gives  $v_i$  the value  $c_i$ .

**Definition 4** Let  $\phi$  be a monotone formula,  $\vec{G} = (G_{01}, G_{11}, \dots, G_{0m}, G_{1m})$  a  $2m$ -tuple of graphs and  $\vec{H}$  an  $m$ -tuple of graphs. We say that  $\vec{H}$  is  $(\phi, \vec{G})$ -i-opened as  $b$  if for each  $i$  an isomorphism is shown between  $H_i$  and  $G_{c_i i}$  and  $C^{\text{iso}}(c_1, \dots, c_m; \phi) = b$ .

The following procedure *I-Label* is used by the prover of our proof system. *I-Label*, on input  $\phi$ ,  $\vec{G}$  and a bit  $a$ , computes a  $m$ -tuple  $\vec{H}$  such that  $\vec{H}$  can be  $(\phi, \vec{G})$ -i-opened as  $a$ .

**Procedure** *I-Label*( $\phi, \vec{G}, a$ ).

1. If  $\phi$  consists of one literal  $v_i$  then set  $H_i$  equal to a randomly chosen graph isomorphic to  $G_{ai}$  and return.
2. If  $\phi = \phi_1 \vee \phi_2$  then randomly choose  $b \in \{0, 1\}$  and execute *I-Label*( $\phi_1, \vec{G}_1, b$ ) and *I-Label*( $\phi_2, \vec{G}_2, a \oplus b$ ) (where  $\vec{G}_1$  and  $\vec{G}_2$  are the tuples of graphs in  $\vec{G}$  corresponding to the literals which occur in  $\phi_1$  and  $\phi_2$ , respectively). Return.
3. If  $\phi = \phi_1 \wedge \phi_2$  then execute *I-Label*( $\phi_1, \vec{G}_1, a$ ) and *I-Label*( $\phi_2, \vec{G}_2, a$ ) ( $\vec{G}_1$  and  $\vec{G}_2$  are as above). Return.

The following lemmas are used to prove the completeness and the soundness of (P,V).

**Lemma 1** Let  $\vec{H}$  be an  $m$ -tuple output by *I-Label* on input  $\phi, \vec{G}$  and  $a$ . Then  $\vec{H}$  can be  $(\phi, \vec{G})$ -i-opened as  $a$ .

**Lemma 2** Let *I-Label-Out*( $\phi, \vec{G}, a$ ) be the probability space generated by the output of *I-Label* on input  $(\phi, \vec{G}, a)$ . If  $(\phi, \vec{G}) \in \Phi(\text{GI})$  then the probability spaces *I-Label-Out*( $\phi, \vec{G}, 0$ ) and *I-Label-Out*( $\phi, \vec{G}, 1$ ) are identical.

**Lemma 3** Let  $\phi$  be a monotone formula and  $\vec{G}$  a  $2m$ -tuple of graphs. If  $(\phi, \vec{G}) \notin \Phi(\text{GI})$ , then for each  $m$ -tuple of graphs  $\vec{H}$  there exists at most one  $b \in \{0, 1\}$  such that  $\vec{H}$  can be  $(\vec{G}, \phi)$ - $i$ -opened as a  $b$ .

We are now ready to present our proof system for  $\Phi(\text{GI})$ .

The proof system (P,V) for $\Phi(\text{GI})$ .
<b>Input:</b> $(\phi, \vec{G})$ .
<b>P.1</b> Randomly choose $a \in \{0, 1\}$ and run procedure <i>I-Label</i> on input $\phi, \vec{G}$ and $a$ obtaining the $m$ -tuple $\vec{H} = (H_1, \dots, H_m)$ . Send $\vec{H}$ to $V$ .
<b>V.1</b> Randomly choose $b \in \{0, 1\}$ and send it to $P$ .
<b>P.2</b> $(\phi, \vec{G})$ - $i$ -open $\vec{H}$ as $b$ .
<b>V.2</b> Verify that for each $i$ there exists $c_i \in \{0, 1\}$ such that an isomorphism is given between $H_i$ and $G_{c_i}$ and that $C^{\text{iso}}(c_1, \dots, c_m; \phi) = b$ .

Let us now prove that the above protocol is indeed a perfect zero-knowledge proof system.

**Theorem 1**  $(P, V)$  is a perfect zero-knowledge proof system for  $\Phi(\text{GI})$ .

**Proof's sketch:** The completeness and the soundness properties are easily derived from Lemmas 1, 2, 3. The perfect zero-knowledge property is proved by exhibiting a simulator whose output has the same distribution of the view of the verifier. The simulator strategy is very simple; it chooses a bit  $b$  at random and constructs  $\vec{H}$  using *I-Label* on input  $\phi, \vec{G}$  and  $b$ . If the verifier asks for  $b$  then the simulator  $(\phi, \vec{G})$ - $i$ -opens  $\vec{H}$ ; otherwise it rewinds the verifier and starts again. ■

We note that Eyal Kushilevitz has independently discovered a protocol for the OR of graph isomorphism.

**Extensions.** The proof system of this section is also a proof system of knowledge. In fact, the extractor can play the role of the verifier in a first execution of the protocol (P,V) by sending a question  $b$ ; then it can rewind the prover to the state immediately after step P.1, and change the question to  $1 - b$ . From the two answers obtained by P at step P.2 in the two different execution of the protocol, the extractor can compute some isomorphisms between input graphs which represent exactly the private input needed by P in order to run the protocol. Moreover, the proof system (P,V) can be modified to require only a constant number of rounds. The procedure *I-Label* can be used by the verifier

to commit to its random bits (that is the value of  $b$  for each round). The simulator runs the verifier twice so to learn the bits  $b$  and then prepares the  $\vec{H}$ 's so to satisfy the verifier.

### 3.2 A perfect zero-knowledge proof system for $\Phi(\text{GNI})$

In this section we present a proof system for monotone formulae closure over graph nonisomorphism. The proof system of the previous section can be seen as a “meet-the-challenge” game (extending and inspired by the original quadratic-residuosity and graph-isomorphism proofs [34, 30]). The prover constructs the  $m$ -tuple  $\vec{H}$  and the verifier challenges him/her to open  $\vec{H}$  as  $b$ . If the prover meets the challenge then the verifier accepts the input. Our proof system for graph nonisomorphism languages, instead, can be seen as a “guessing” game (as in the original quadratic non-residuosity and graph non-isomorphism proofs [34, 30]). More precisely, the verifier now constructs  $\vec{H}$  such that if the formula is true, then  $\vec{H}$  can be opened in exactly one way; on the other hand if the formula is false then each  $m$ -tuple  $\vec{H}$  can be opened both as 0 and as 1. The verifier accepts the input if the prover guesses correctly which way it can be opened. For the zero-knowledge property we have to make sure that  $\vec{H}$  has been constructed properly (that is  $\vec{H}$  can be opened as a bit).

Let us now proceed more formally and introduce the notions of non-isomorphism-value and of  $n$ -opening of an  $m$ -tuple. Roughly speaking, we say that if  $\phi = \phi_1 \wedge \phi_2$  then the non-isomorphism value of  $\phi$  is the ex-or between those of  $\phi_1$  and  $\phi_2$ , while if  $\phi = \phi_1 \vee \phi_2$ , and the non-isomorphism values of  $\phi_1$  and  $\phi_2$  are equal, then the non-isomorphism value of  $\phi$  is equal to this value; otherwise it is set equal to the undefined value  $-$ . The notion of  $(\phi, \vec{G})$ - $n$ -opening an  $m$ -tuple  $\vec{H}$  as  $b$  is the same as  $i$ -opening except that we require the non-isomorphism-value (instead of isomorphism-value) to be  $b$ . In the description of our proof system we use the procedure *N-Label*. *N-Label* is similar to *I-Label* except that the conditions checked at steps 2 and 3 are

swapped.

**Procedure**  $N\text{-Label}(\phi, \vec{G}, a)$ .

1. If  $\phi$  consists of a single literal  $v_i$  then set  $H_i$  equal to a randomly chosen graph isomorphic to  $G_{a_i}$  and return.
2. If  $\phi = \phi_1 \wedge \phi_2$  then randomly choose  $b \in \{0, 1\}$  and execute  $N\text{-Label}(\phi_1, \vec{G}_1, b)$  and  $N\text{-Label}(\phi_2, \vec{G}_2, a \oplus b)$ . Return.
3. If  $\phi = \phi_1 \vee \phi_2$  then execute  $N\text{-Label}(\phi_1, \vec{G}_1, a)$  and  $N\text{-Label}(\phi_2, \vec{G}_2, a)$ . Return.

o

Procedure  $N\text{-Label}$  enjoys properties similar to  $I\text{-Label}$  that are summarized in the following lemmas.

**Lemma 4** *Let  $\vec{H}$  be an output of  $N\text{-Label}$  on input  $\phi, \vec{G}$  and  $b$ . If  $(\phi, \vec{G}) \in \Phi(\text{GNI})$  then  $\vec{H}$  can be  $(\phi, \vec{G})$ -opened as  $b$  but not as  $1 - b$ .*

**Lemma 5** *Let  $\phi$  be a monotone formula on  $m$  variables,  $\vec{G}$  a  $2m$ -tuple of graphs and  $N\text{-Label-Out}(\phi, \vec{G}, a)$  the probability space generated by the output of  $N\text{-Label}$  on input  $(\phi, \vec{G}, a)$ . If  $(\phi, \vec{G}) \notin \Phi(\text{GNI})$  then the probability spaces  $N\text{-Label-Out}(\phi, \vec{G}, 0)$  and  $N\text{-Label-Out}(\phi, \vec{G}, 1)$  are identical.*

In the description of our proof system, we use the procedure  $Check$  that is described below. It takes as inputs  $\vec{G}, \vec{H}, \phi$  and a bit  $a$  and outputs  $\vec{L}$  such that  $\vec{H}$  can be  $(\phi, \vec{L})$ -n-opened as  $a$  and, for each  $j$ ,  $(L_{0j}, L_{1j}) \approx (G_{0j}, G_{1j})$ ; that is, either  $L_{0j} \approx G_{0j}$  and  $L_{1j} \approx G_{1j}$  or  $L_{0j} \approx G_{1j}$  and  $L_{1j} \approx G_{0j}$ . These two conditions imply that  $\vec{H}$  can be  $(\phi, \vec{G})$ -n-opened. The  $2m$ -tuple  $\vec{L}$  is used by the verifier to convince the prover that  $\vec{H}$  is well constructed in the following way. First, using  $Check$ , the verifier constructs the  $2m$ -tuples  $\vec{L}^1, \dots, \vec{L}^m$ . Then, for  $i = 1, \dots, m$ , the prover asks the verifier either to show that, for each  $j$ ,  $(L_{0j}^i, L_{1j}^i) \approx (G_{0j}, G_{1j})$ , or to  $(\phi, \vec{L}^i)$ -n-open  $\vec{H}$ . If all its requests are satisfied, the prover is confident that the verifier knows how to  $(\phi, \vec{G})$ -n-open  $\vec{H}$ . Roughly speaking, if this were not the case, then, for each  $i$ , the verifier could only answer at most one of the two possible questions.

We are now ready to present our proof system.

**The Proof System**  $(A, B)$  for  $\Phi(\text{GNI})$

**Input:**  $(\phi, \vec{G})$ .

**B.1** Randomly choose  $b \in \{0, 1\}$ .

Run procedure  $N\text{-Label}$  on input  $\phi, \vec{G}, b$  thus obtaining  $\vec{H}$  and send  $\vec{H}$  to A.

For  $i = 1, \dots, mn^2$

    Randomly choose  $a_i \in \{0, 1\}$ .

    Run procedure  $Check$  on input  $\phi, a_i, \vec{G}, \vec{H}$  obtaining the  $2m$ -tuple  $\vec{L}^i$  and send  $\vec{L}^i$  to A.

**A.1** For  $i = 1, \dots, mn^2$ , randomly choose  $c_i \in \{0, 1\}$  and send it to B.

**B.2** For  $i = 1, \dots, mn^2$

    If  $c_i = 0$  then

        for  $j = 1, \dots, m$

            Let  $b_{ij}$  be such that  $G_{0i} \approx L_{b_{ij}i}^i$  and  $G_{1i} \approx L_{1-b_{ij}i}^i$ .

            Show isomorphisms from  $G_{0j}$  to  $L_{b_{ij}i}^i$

            and from  $G_{1j}$  to  $L_{1-b_{ij}i}^i$ .

    If  $c_i = 1$  then

$(\phi, \vec{L}^i)$ -n-open  $\vec{H}$  as  $a_i$ .

**A.2** Verify that step B.2 has been correctly performed. Let  $a'$  be such that  $\vec{H}$  can be  $(\phi, \vec{G})$ -n-opened.

Send  $a'$  to B.

**B.3** If  $a = a'$  then ACCEPT. Else REJECT.

**Procedure**  $Check(\phi, a, \vec{G}, \vec{H})$ .

1. If  $\phi = v_i$  then

    If  $H_i \approx G_{0i}$  then

        set  $L_{ai}$  to a random isomorphic copy of  $G_{0i}$

        set  $L_{1-a,i}$  to a random isomorphic copy of  $G_{1i}$

    else

        set  $L_{ai}$  to a random isomorphic copy of  $G_{1i}$

        set  $L_{1-a,i}$  to a random isomorphic copy of  $G_{0i}$

    Return.

2. If  $\phi = \phi_1 \vee \phi_2$  then

    execute  $Check(\phi_1, a, \vec{G}_1, \vec{H}_1)$  and  $Check(\phi_2, a, \vec{G}_2, \vec{H}_2)$ .

    Return.

3. If  $\phi = \phi_1 \wedge \phi_2$  then

    randomly choose  $b \in \{0, 1\}$  and execute

$Check(\phi_1, b, \vec{G}_1, \vec{H}_1)$  and  $Check(\phi_2, a \oplus b, \vec{G}_2, \vec{H}_2)$ .

    Return.

The proof of the following lemma is obvious.

**Lemma 6** *Let  $\vec{L}$  be an output of the procedure  $Check$  on input  $\phi, a, \vec{G}$  and  $\vec{H}$ . Then  $\vec{H}$  can be  $(\phi, \vec{L})$ -opened as  $a$ . Moreover, for each  $1 \leq i \leq m$   $(G_{0i}, G_{1i}) \approx (H_{0i}, H_{1i})$ .*

We are now ready to present the main result of this section.

**Theorem 2**  *$(A, B)$  is a perfect zero-knowledge proof system for  $\Phi(\text{GNI})$ .*

**Proof's sketch:** *Completeness.* If  $(\phi, \vec{G}) \in \Phi(\text{GNI})$  then the verifier can always satisfy the prover's request at step A.2 (see Lemma 6). Moreover, as  $\vec{H}$  has been computed using *N-Label*, it can be opened in exactly one way (see Lemma 4). Therefore the prover can always guess the bit  $b$ .

*Soundness.* Let  $\phi$  be a formula on  $m$  literals and  $\vec{G}$  be a  $2m$ -tuple of graphs such that  $(\phi, \vec{G}) \notin \Phi(\text{GNI})$ . The verifier accepts only if the prover guesses correctly the value of  $a$ , that is, the value with respect to which the verifier is able to  $(\phi, \vec{G})$ -n-open  $\vec{H}$ . However, if  $(\phi, \vec{G}) \notin \Phi(\text{GNI})$  then  $\vec{H}$  can be opened both as a 0 and as a 1 (see Lemmas 4 and 5). Moreover, observe that  $\vec{L}^i$  and the bits  $b_{ij}$  are computed on input values whose distribution is independent from the value of  $a$  (remember that  $\vec{H}$  can be opened both as 0 and as 1). This means that by seeing this the prover has no information on  $a$ .

Therefore, as  $a$  is chosen at random by the verifier, any prover has probability at most  $1/2$  of satisfying the verifier's request.

*Perfect Zero Knowledge.* For the zero-knowledge part we have to exhibit a simulator whose output has the same distribution as the view of the verifier.

The difficult thing here is that the simulator has to guess the bit  $a$ . We do this employing the following strategy. The simulator executes step A.1 twice with different values for the bits  $c_i$ . This means that for each  $\vec{L}$  the simulator sees both the isomorphism between the graphs in  $\vec{G}$  and graphs in  $\vec{L}$  (for  $c_i = 0$ ) and between the graphs in  $\vec{L}$  and the graphs in  $\vec{H}$  (remember that if  $c_i = 1$ ,  $\vec{H}$  is  $(\phi, \vec{L})$ -n-opened). This, with very high probability, gives the simulator isomorphisms between  $\vec{G}$  and  $\vec{H}$  and from this (more precisely, from knowing to which of  $G_{0i}$  and  $G_{1i}$   $\vec{H}_i$  is isomorphic) the value of  $a$  is easily computed. In the event that the simulator is not able to obtain  $a$  we resort to the technique of [30] of running an exponential time algorithm for computing  $a$ . As this event happens with negligible probability the simulator still runs in expected polynomial time. ■

### 3.3 Proof systems for random self reducible languages

The class of random self reducible languages has been introduced in [43, 1] and has been later referred to as RSR.

**RSR languages.** Let  $\mathcal{N}$  be a countably infinite set and, for each  $x \in \mathcal{N}$ , let  $R_x$  be a relation verifiable

in polynomial-time. Define the domain of  $R_x$  as the set  $\{z | \exists w \text{ such that } (z, w) \in R_x\}$ . If  $(z, w) \in R_x$  then we say that  $w$  is an  $x$ -witness for  $z$ . The language  $L_R = \{(x, z) | \exists w \text{ such that } (z, w) \in R_x\}$  is random self reducible (RSR) if there exists a polynomial time algorithm  $\text{Sample}_{L_R}$  that, on input  $x, z, r$  outputs  $y$  such that  $(x, y) \in L_R$  and

1. If  $r$ 's bits are random, uniform, and independent then  $y$  is uniformly distributed over the domain of  $R_x$ .

2. There exists a polynomial-time algorithm  $A_1$  ( $A_2$ ) that, on input  $x, y, r$  and an  $x$ -witness for  $y$  (for  $z$ ) outputs an  $x$ -witness for  $z$  (for  $y$ ).

3. There exists an algorithm  $\text{Generate}_{L_R}$  that, on input  $x$ , outputs in expected polynomial time a pair  $(z, w) \in R_x$  with  $z$  uniformly distributed over the domain of  $R_x$  and  $w$  uniformly distributed over all  $x$ -witnesses for  $z$ .

Known examples of random self reducible languages include the language of graph isomorphism, the language of quadratic residues, and the language of multiplicative subgroup of  $Z_p^*$  generated by  $g$  ( $p$  a prime). In [43], it is proved that all language in RSR have PZK proof systems. In the full version we show that the same holds for co-RSR.

The proof system for  $\Phi(\text{GI})$  can be readily adapted to obtain a proof system for  $\Phi(L)$ , where  $L$  is a general random self reducible language. The prover on input a formula  $\phi$  on  $m$  literals and a  $m$ -tuple of pairs  $\vec{XZ} = ((x_1, z_1), \dots, (x_m, z_m))$  constructs, using the procedure  $\text{GEN-I-Label}_L$  described below, an  $m$ -tuple  $\vec{C} = (c_1, \dots, c_m)$ . If  $(\phi, \vec{XZ}) \in \Phi(L)$ , then the output of  $\text{GEN-I-Label}_L$  on input  $(\phi, \vec{XZ}, 0)$  and the output of  $\text{GEN-I-Label}_L$  on input  $(\phi, \vec{XZ}, 1)$  have the same distribution. On the other hand if  $(\phi, \vec{XZ}) \notin \Phi(L)$  then all  $m$ -tuples can be opened in at most one way. Then the verifier chooses  $b$  at random and asks to open  $\vec{C}$  as  $b$ . The verifier accepts if the prover satisfies its request.

**Procedure**  $\text{GEN-I-Label}_L(\phi, \vec{XZ}, a)$ .

1. If  $\phi$  consists of literal  $v_i$  then randomly choose  $r$ ; if  $a = 0$  then run  $\text{Sample}_{L_R}(x_i, z_i, r)$  obtaining  $y_i$  and set  $c_i = y_i$ ; if  $a = 1$  then run  $\text{Generate}_{L_R}(x_i)$  obtaining  $y_i, w_i$  and set  $c_i = y_i$ ;
2. If  $\phi = \phi_1 \vee \phi_2$  then randomly choose  $b \in \{0, 1\}$  and execute  $\text{GEN-I-Label}_L(\phi_1, \vec{XZ}, b)$  and  $\text{GEN-I-Label}_L(\phi_2, \vec{XZ}, a \oplus b)$ . Return.
3. If  $\phi = \phi_1 \wedge \phi_2$  then execute  $\text{GEN-I-Label}_L(\phi_1, \vec{XZ}, a)$  and  $\text{I-Label}(\phi_2, \vec{XZ}, a)$ . Return.

Thus we obtain the following:

**Theorem 3** *Let  $L$  be a random self reducible language. Then  $\Phi(L) \in \text{PZK}$ .*

A similar extension of procedure *N-Label* and of the proof system for  $\Phi(\text{GNI})$  yields

**Theorem 4** *Let  $L$  be a co-RSR language. Then  $\Phi(L) \in \text{PZK}$ .*

### 3.4 Extensions

**Non-Monotone Formulae: mixing GI and GNI** We now describe proof systems for certain non-monotone formulae, namely, where there are statements both on isomorphism and nonisomorphism. We explain the protocol idea on a very simple example and then discuss the extension to a more complex set of cases.

*The basic proof system.* Consider the language GI-OR-GNI of quadruples of graphs  $(G_0, G_1, H_0, H_1)$  such that  $(G_0 \not\approx G_1) \vee (H_0 \approx H_1)$ . The following is a perfect zero-knowledge proof system for GI-OR-GNI. The first to become active is the verifier that constructs a graph  $G$  isomorphic to one of  $G_0$  and  $G_1$  (in this way it has “committed” to a bit  $b$ ). Then the prover presents the verifier with another graph  $H$  (how the graph  $H$  is constructed is explained in the discussion below). The verifier now shows an isomorphism between  $G$  and  $G_b$  thus asking the prover to show an isomorphism between  $H$  and  $H_b$ . The verifier accepts if the prover satisfies its request.

*An informal discussion.* Let us convince ourselves that the above is indeed a proof system. Suppose that the quadruple does belong to GI-OR-GNI. If the  $G_i$ 's are non isomorphic then the prover by seeing the graph  $G$  can read the bit  $b$  and thus prepare  $H$  isomorphic to  $H_b$  so to satisfy the verifier's request. Suppose now that the  $G_i$ 's are isomorphic and thus the prover does not learn  $b$  in advance. However, in this case the  $H_i$ 's are isomorphic and thus for the prover it is enough to construct  $H$  isomorphic to  $H_1$  and then, after  $b$  is revealed by the verifier, to show an isomorphism between  $H$  and  $H_b$ . The soundness property is argued in a similar way. If the quadruple does not belong to GI-OR-GNI then, after receiving  $G$  from the verifier, the prover has no clue about the value of  $b$ . Moreover, the graph  $H$  it constructs is isomorphic to at most one of  $H_0$  and  $H_1$ . Therefore, the verifier accepts with probability at most  $1/2$ . For the perfect zero-knowledge we need to add a step in which, similarly to what done in Section 3.2, the verifier shows that

it has constructed the graph  $G$  correctly. This allows the simulator to learn the value of  $b$  and to construct  $H$  appropriately.

**Theorem 5** *The language GI-OR-GNI has a perfect zero-knowledge proof system.*

*The generalized OR.* The technique just presented can be used to obtain proof systems for the OR of two composite (monotone formula) statements: one about isomorphism and the other about non isomorphism. Roughly speaking, the non-isomorphism part is used by the verifier to commit to a bit  $b$  using the procedure *N-Label*. Then using the procedure *I-Label* the prover constructs a vector  $\vec{H}$ . Then the verifier shows the value  $b$  and the prover has to open  $\vec{H}$  as a  $b$ .

**A proof system for threshold formulae.** A threshold formula is a formula built using threshold gates. A  $(k, l)$ -gate is a  $k$ -input 1-output gate; its output is 1 iff at least  $l$  inputs are 1. AND and OR gates are special cases of thresholds and thresholds do have polysize monotone boolean formulae [44]. Thus, using the proof system for boolean monotone formulae, we could obtain a proof system for the monotone formula of size  $O(n^{5.3})$  for the threshold function given by Valiant in [44]. We show however a much more efficient construction that is constructive (Valiant's proof that thresholds have monotone formulae does not provide an effective construction).

The proof system is also based on the concept of “opening” but we use secret sharing. Let us explain the main idea for the case of a single  $(k, m)$  threshold gate. We have in input  $\vec{G}$  and want to prove that at least  $k$  out of the  $m$  pairs of graphs consist of isomorphic graphs (again we are presenting the result in terms of graph isomorphism but it can be easily extended to any RSR (or co-RSR) language). Prover and verifier agree upon a  $(m-k+1, m)$  threshold secret sharing scheme (e.g., Shamir's scheme [41]) and, for each  $i$ , the prover constructs  $l = \lfloor \log m \rfloor + 1$  graphs  $H_{1i}, \dots, H_{li}$ . The verifier replies by sending a bit  $b$ . Then, for each  $H_{ij}$ , the prover shows an isomorphism between  $H_{ij}$  and  $G_{b_{ij}i}$  in such a way that the following condition is met. For each  $i$ , let  $x_i$  be the integer with binary representation  $b_{i1} \dots b_{il}$ ; then  $x_1, \dots, x_m$  are a sharing of the bit  $b$ .

Now, why is this a proof system? The main observation here is that if  $G_{i0} \approx G_{i1}$  then  $x_i$  can be chosen by the prover after it has seen  $b$ . On the

other hand if  $G_{i_0} \not\approx G_{i_1}$  then  $x_i$  is fixed and cannot be changed by the prover after  $b$  has been given to him. Suppose that the statement is true. This means that no more than  $m - k$  pairs are non-isomorphic and thus no more than  $m - k$   $x_i$ 's are fixed. But in this case, the remaining  $x_i$ 's (at least  $k$  of them) can always be chosen so that the  $x_i$ 's constitute a sharing of  $b$ . Suppose instead that the non-isomorphic pairs are  $m - k + 1$ . Then, the corresponding values by themselves determine the value of the secret and this value will be equal to  $b$  with probability  $\leq 1/2$ . The perfect zero-knowledge property is easily proved by using the trial-and-error strategy.

The technique used to give a proof system for one gate can be iterated to obtain a proof system for any threshold formula. Erez Petrank [40] has shown how our technique can be used to obtain proof systems for symmetric functions. In a preliminary work on the non-interactive model, the specific language of quadratic residuosity modulo special (Blum) integers was discussed in [20] (inspired by a preliminary version of the work on the RSR and co-RSR interactive protocol results [22]). In particular, a statistical zero-knowledge proof system for one single threshold gate was given; that result does not generalize to formulae. Also, in [19], related ideas of using threshold schemes based proofs for generating "witness hiding proofs of knowledge" were presented.

## 4 Closure in the shared-string model

In this section we consider the non-interactive model of [12, 13] for SZK and derive results about SZK in the interactive model. We prove that if a language  $L$  has a non-interactive SZK proof system then honest-verifier interactive SZK proof systems exist for all monotone boolean formulae whose atoms are statements about the complement of  $L$ . Let us review the definition of non-interactive statistical zero-knowledge proof system [12].

**Definition 5** A pair  $(A, B)$ , where  $A(\cdot, \cdot)$  is a probabilistic Turing machine and  $B(\cdot, \cdot, \cdot)$  is a deterministic Turing machine running in time polynomial in the length of its second input, is a non-interactive proof system for the language  $L$  if there exists a constant  $c > 0$  such that

1. (Completeness) For all sufficiently long  $x \in L$

$$\Pr(B(\sigma, x, A(\sigma, x)) = \text{ACCEPT}) > 1 - 2^{-|x|^c},$$

where the probability is taken over the random choices of  $\sigma \in \{0, 1\}^{|x|^c}$  and  $A$ 's coin tosses.

2. (Soundness) For all algorithms  $A^*$ , and all sufficiently long  $x \notin L$

$$\Pr(B(\sigma, x, A^*(\sigma, x)) = \text{ACCEPT}) < 2^{-|x|^c},$$

where the probability is taken over the random choices of  $\sigma \in \{0, 1\}^{|x|^c}$  and  $A$ 's coin tosses.

We call  $\sigma$  the reference string.

We define  $View(x)$ , the verifier's view of  $A$ 's proof on input  $x$ , as the probability space that assigns to pairs  $(\sigma; Proof)$  the probability that  $\sigma$  is the reference string and that  $Proof$  is the output of  $A$  on input  $\sigma$  and  $x$ .

**Definition 6** A non-interactive proof system  $(A, B)$  for  $L$  is a non-interactive statistical zero-knowledge proof system for  $L$  if there exists a probabilistic polynomial-time machine  $M$  such that for all constants  $d$  and all sufficiently long  $x \in L$

$$\sum_{\alpha} |\Pr(View(x) = \alpha) - \Pr(M(x) = \alpha)| < |x|^{-d}.$$

For sake of a better understanding, we first present a simple result about the complement that contains the basic ideas needed for the extension to monotone formulae. Then we briefly discuss the extensions needed to obtain honest-verifier SZK proof systems for all monotone formulae over  $\bar{L}$ .

### 4.1 A special bit commitment scheme

Our results are based on a special type of bit commitment for honest committer that is constructed using non-interactive SZK proofs. A *bit commitment scheme based on the language  $L$* , just as regular bit commitment schemes, is a game between two players sender and receiver and consists of two phases: a commitment phase and a recovering phase. In the commitment phase the sender, on input a random bit  $b$  to be committed and an auxiliary string  $x$ , computes in expected polynomial time a commitment key  $com$  and sends the pair  $(x, com)$  to the receiver. The pair  $(x, com)$  has the following property: 1) if  $x \in L$ , then no adversary can guess the committed bit  $b$  significantly better than guessing at random; 2) if  $x \notin L$ , on input  $x$  and  $com$ , then it is possible to guess  $b$  correctly with overwhelming probability, even though this might require more than polynomial time.

Let us show how to construct a bit commitment scheme  $BC_L$  based on a language  $L$  that has a non-interactive SZK proof system. We denote by  $(P,V)$  the noninteractive SZK proof system for  $L$  and let  $p(n)$  denote the length of the reference string required by  $(P,V)$  for inputs of length  $n$ . On input  $b$ , the bit to be committed to, and an auxiliary string  $x$  the sender performs the following instructions. If  $b = 0$  then the sender chooses a random string  $\sigma$  of length  $p(|x|)$  and sets  $com$  equal to  $\sigma$ . On the other hand, if  $b = 1$  then the sender runs the simulator  $S$  for  $(P,V)$  on input  $x$ , obtaining  $(\sigma, Proof)$ . If  $V(\sigma, x, Proof) = \text{ACCEPT}$  then the sender sets  $com = \sigma$ . On the other hand if  $V(\sigma, x, Proof) \neq \text{ACCEPT}$  then the sender sets  $com = \emptyset$ .

Now if  $x \in L$  then, since  $S$  is a statistical zero-knowledge simulator for  $(P,V)$ , the string  $\sigma$  produced by  $S$  is statistically indistinguishable from a truly random string of length  $p(|x|)$ . Therefore, property 1) above holds.

Suppose now that  $x \notin L$ . If  $b = 0$ , then  $\sigma$  is a random string of length  $p(|x|)$ . Suppose  $b = 1$  and distinguish two cases depending on whether the verifier accepts or rejects the output of the simulator. If the simulator has output an accepting pair  $(\sigma, Proof)$ , then, by the soundness of  $(P,V)$ , the string  $\sigma$  belongs to the exponentially small set of strings of length  $p(|x|)$  that allow a cheating prover to fool  $V$ . Therefore, in this case, the receiver can distinguish  $\sigma$  from a truly random string and guess  $b$ . If the simulator has output a rejecting pair then trivially the receiver can distinguish the two cases. Therefore we can conclude that also property 2) holds.

## 4.2 Honest-verifier SZK proof systems

Let  $L$  be a language with a non-interactive SZK proof system. We start by describing a honest-verifier SZK proof system for  $\bar{L}$ .

A honest-verifier zero-knowledge proof system  $(C,D)$  for proving that a string  $x \in \bar{L}$  is immediately obtained from the bit commitment  $BC_L$  in the following way.  $D$  uniformly chooses a bit  $b$  and runs the program of the sender on input  $x$  and  $b$  thus obtaining a commitment key  $com$ . The prover  $C$  now reads  $com$  and send its guess for the bit  $b$ . If it succeeds then  $D$  accepts otherwise  $D$  rejects.

Completeness and soundness follow from properties 2 and 1, respectively, of the bit commitment  $BC_L$  (for the soundness we have to repeat the pro-

ocol). For the zero-knowledge property, the simulator strategy is very simple: it uniformly chooses a bit  $b$  and constructs the commitment key  $com$  by running the program of the sender of the bit commitment  $BC_L$  using the string  $R$  as source of random bits. The output of the simulator is then  $(b, R; com, b)$  whose distribution is equal to the view of the verifier up to a negligible factor. We can now state the following theorem (given also in [8]):

**Theorem 6** *If  $L$  has a non-interactive SZK proof system then  $\bar{L}$  has a honest-verifier SZK proof system.*

In order to obtain honest-verifier SZK proof systems for all monotone formulae we need to generalize the commitment scheme  $BC_L$ . In particular, we want to construct a commitment scheme that takes as input a monotone boolean formula  $\phi$  on  $m$  literals, a random bit  $b$  to be committed, and an  $m$ -tuple  $\vec{XZ} = (x_1, \dots, x_m)$  of strings and outputs an  $m$ -tuple of keys  $(com_1, \dots, com_m)$  with the following properties:

- 1) if  $(\phi, \vec{XZ}) \in \Phi(\bar{L})$  then it is possible to recover the bit  $b$  from  $\vec{XZ}$  and  $(com_1, \dots, com_m)$  (even though this might require super-polynomial time).
- 2) if  $(\phi, \vec{XZ}) \notin \Phi(\bar{L})$  then no adversary on input  $\vec{XZ}$  and  $(com_1, \dots, com_m)$  can guess the bit  $b$  with probability significantly better than  $1/2$ .

We achieve this by using the following generalization of procedure  $N$ -Label of Section 3.2.

**Procedure**  $GenBC_L(\phi, \vec{XZ}, a)$ .

1. If  $\phi$  consists of a single literal  $v_i$  then run  $BC_L$  on input  $x_i$  and  $a$  obtaining  $com_i$ . Return.
2. If  $\phi = \phi_1 \wedge \phi_2$  then randomly choose  $b \in \{0,1\}$  and execute  $GenBC_L(\phi_1, \vec{XZ}_1, b)$  and  $GenBC_L(\phi_2, \vec{XZ}_2, a \oplus b)$ . Return.
3. If  $\phi = \phi_1 \vee \phi_2$  then execute  $GenBC_L(\phi_1, \vec{XZ}_1, a)$  and  $GenBC_L(\phi_2, \vec{XZ}_2, a)$ . Return.

We can thus state the main result of this section:

**Theorem 7** *If the language  $L$  has a non-interactive SZK proof system, then the language  $\Phi(\bar{L})$  has a honest-verifier SZK proof system.*

From the result of [39] (following the model of [6]), it follows that:

**Corollary 1** *Let  $L$  be a language with a non-interactive SZK proof system. If one-way permutations exist,  $\Phi(\bar{L})$  has a SZK proof system.*

Let  $L$  and  $L'$  be two languages and define  $OR(L, L')$  as the language of pairs  $(x_1, x_2)$  such

that either  $x_1 \in L$  or  $x_2 \in L'$ . We next describe a honest-verifier zero-knowledge proof system  $(P, V)$  for  $\text{OR}(L_1, \overline{L_2})$  for the case in which both  $L_1$  and  $L_2$  have non-interactive SZK proof systems that we denote by  $(P_1, V_1)$  and  $(P_2, V_2)$ , respectively. On input  $(x_1, x_2)$ , the first to become active is  $V$  that uniformly chooses a reference string  $\sigma_V$  for  $(P_1, V_1)$  and commits to it using the bit commitment scheme  $BC_{L_2}$ . The prover after receiving the commitment from  $V$  behaves differently according to whether  $x_2$  belongs or not to  $L_2$ . If  $x_2 \notin L_2$  then  $P$  reads the string  $\sigma_V$ ; runs the simulator  $S_1$  for  $L_1$  on input  $x_1$  obtaining  $(\sigma_P, \text{Proof})$  and sets  $\sigma = \sigma_P \oplus \sigma_V$ . If  $x_2 \in L_2$  then  $P$  uniformly chooses a reference string  $\sigma$ . In both cases it sends  $\sigma$  to  $V$  that answers by sending  $\sigma_V$ . Finally  $P$  sets  $\tau = \sigma \oplus \sigma_V$  and produces a non-interactive proof that  $x_1 \in L_1$  relative to the reference string  $\tau$ .

Now we give a sketch of proof that  $(P, V)$  is a honest-verifier zero-knowledge proof system for  $\text{OR}(L_1, \overline{L_2})$ . For the completeness property, observe that if  $x_1 \in L_1$  then  $\tau$  is chosen at random, and thus  $P$  can convince with high probability that  $x_1 \in L_1$  (this follows from the completeness of the proof system for  $L_1$ ). If  $x_2 \in \overline{L_2}$  then  $P$  can compute the string  $\sigma_V$  committed by  $V$  (thanks to the properties of the bit commitment scheme used) and thus sets  $\sigma = \sigma_P \oplus \sigma_V$  so that  $\tau$  turns out to be equal to  $\sigma_P$  and it can give the verifier *Proof*.

For the soundness property, observe that if  $x_1 \notin L_1$  and  $x_2 \in L_2$ , then any prover guesses correctly the string  $\sigma_V$  only with negligible probability. Then, however it chooses  $\sigma$ , the string  $\tau$  is uniformly chosen and thus can compute a convincing *Proof* only with negligible probability (by the soundness of the proof system for  $L_1$ ).

For the zero-knowledge property, the simulator strategy is the following: it runs the simulator  $S_1$  on input  $x_1$  and obtains  $(\sigma_P, \text{Proof})$  as output. Then it randomly chooses a reference string  $\sigma_V$  and computes  $\sigma = \sigma_P \oplus \sigma_V$ . Then it constructs the commitment key *com* for the string  $\sigma_V$  by running the program of the sender of the bit commitment  $BC_{L_2}$  using a randomly chosen string  $R$  as source of random bits. Finally it outputs  $(\sigma_V, R; \text{com}, \sigma, \sigma_V, \text{Proof})$ .

Finally we observe that we can use the same techniques of the proof system  $(A, B)$  for  $\Phi(\text{GNI})$  to obtain a proof system for  $\text{OR}(L_1, \Phi(\overline{L_2}))$  for any monotone formula  $\Phi$ .

## 5 Open problems

This work proposes several new open problems. It would be interesting to obtain statistical zero-knowledge proof systems for general non-monotone formulae or to prove related limitations; in 3.4, we suggest specific useful formula structure for which closure applies, demonstrating a case where closure is possible. Can this be extended? It would be also interesting to generalize the results to other protocol techniques yielding closure of more general classes of interactive SZK languages. Perhaps, one can exploit randomness in protocols as we employed the random string in the non-interactive SZK case.

**Acknowledgment:** We would like to thank Erez Petrank for very useful discussions, and Mihir Bellare, Oded Goldreich and Johan Håstad for their remarks, corrections and suggestions on an early manuscript.

## References

- [1] M. Abadi, J. Feigenbaum, and J. Kilian *On Hiding Information from an Oracle*, in Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, pp. 195–203.
- [2] W. Aiello and J. Håstad. *Statistical Zero Knowledge Can Be Recognized in Two Rounds* J. Comput. System Sci. 42, 1991, pp. 327–345.
- [3] L. Babai and S. Moran, *Arthur–Merlin Games: A Randomized Proof System and a Hierarchy of Complexity Classes*, Journal of Computer and System Sciences, vol. 36, 1988, pp. 254–276.
- [4] M. Bellare and O. Goldreich, *On Defining Proofs of Knowledge*, in Proc. of CRYPTO '92.
- [5] M. Bellare, S. Micali, and R. Ostrovsky, *Perfect Zero Knowledge in Constant Rounds*, in Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, 1990, pp. 482–493.
- [6] M. Bellare, S. Micali, and R. Ostrovsky. *The (True) Complexity of Statistical Zero Knowledge*, in Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, 1990, pp. 494–502.
- [7] M. Bellare and E. Petrank. *Making Zero-Knowledge Provers Efficient*. In Proceedings of the 24th Annual ACM Symposium on Theory of Computing, 1992, pp. 711–722.
- [8] M. Bellare and P. Rogaway, *Perfect Non-Interactive Zero-Knowledge*, Unpublished Manuscript, 1990.
- [9] M. Bellare and M. Yung, *Certifying Cryptographic Tools: The Case of Trapdoor Permutations*, in Proc. of CRYPTO 92.
- [10] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, S. Micali, and P. Rogaway, *Everything Provable is Provable in Zero Knowledge*, in Proc. of CRYPTO 88.

- [11] M. Blum, *Program Result Checking: A New Approach to Making Programs More Reliable*, Proceedings of 20th ICALP, 1993, pp. 1–14.
- [12] M. Blum, A. De Santis, S. Micali, and G. Persiano, *NonInteractive Zero-Knowledge*, SIAM Journal of Computing, vol. 20, no. 6, Dec. 1991, pp. 1084–1118.
- [13] M. Blum, P. Feldman, and S. Micali, *Non-Interactive Zero-Knowledge and Applications*, in Proceedings of the 20th Annual ACM Symposium on Theory of Computing, 1988, pp. 103–112.
- [14] M. Blum and S. Kannan, *Designing Programs that Check their Work*, in Proceedings of the 21st Annual ACM Symposium on Theory of Computing, 1989, pp. 86–97.
- [15] J. Boyar, K. Friedl, and C. Lund, *Practical Zero-Knowledge Proofs: Giving Hints and Using Deficiencies*, J. of Cryptology, n. 4, pp. 185–206, 1991.
- [16] R. Boppana, J. Håstad, and S. Zachos, *Does  $co-NP$  have Short Interactive Proofs?*, Inf. Proc. Lett., vol. 25, May 1987, pp. 127–132.
- [17] G. Brassard, D. Chaum, and C. Crépeau, *Minimum Disclosure Proofs of Knowledge*, Journal of Computer and System Sciences, vol. 37, no. 2, 1988, pp. 156–189.
- [18] G. Brassard, C. Crépeau, and M. Yung, *Perfect Zero-Knowledge Computationally Convincing Proofs for  $NP$  in Constant Rounds*, Theoretical Computer Science, vol. 84, n. 1 (1991) pp. 23–52.
- [19] R. Cramer, I. Damgård, and B. Schoenmakers, *Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols*, in Proc. of CRYPTO 94.
- [20] A. De Santis, G. Di Crescenzo, and G. Persiano, *Secret Sharing and Perfect Zero-Knowledge*, in Proc. of CRYPTO 93.
- [21] A. De Santis, S. Micali, and G. Persiano, *Non-Interactive Zero-Knowledge Proof-Systems*, in Proc. of CRYPTO 87.
- [22] A. De Santis, G. Persiano, and M. Yung, *Formulae over Random Self-Reducible Languages: The Extended Power of Perfect Zero Knowledge*, manuscript, 92.
- [23] U. Feige, A. Fiat, and A. Shamir, *Zero-Knowledge Proofs of Identity*, J. of Cryptology, vol. 1, 1988, pp. 77–94.
- [24] U. Feige, D. Lapidot, and A. Shamir, *Multiple Non-Interactive Zero-Knowledge Proofs Based on a Single Random String*, in Proceedings of the 31st IEEE Symposium on Foundations of Computer Science, 1990, pp. 308–317.
- [25] M. Fischer, S. Micali, and C. Rackoff, *A Secure Protocol for the Oblivious Transfer*, A talk given in Eurocrypt 1984.
- [26] L. Fortnow, *The Complexity of Perfect Zero Knowledge*, in Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, pp. 204–209.
- [27] Z. Galil, S. Haber, and M. Yung, *Minimum-Knowledge Interactive Proofs for Decision Problems*, SIAM Journal on Computing, 1989. (originally: FOCS 85)
- [28] O. Goldreich and H. Krawczyk, *On the Composition of Zero-Knowledge Proof Systems*, Proceedings of 17th ICALP, 1986, pp. 174–187.
- [29] O. Goldreich and E. Kushilevitz, *A Perfect Zero Knowledge Proof for a Decision Problem Equivalent to Discrete Logarithm*, in Proc. of CRYPTO 88.
- [30] O. Goldreich, S. Micali, and A. Wigderson, *Proofs that Yield Nothing but their Validity or All Languages in  $NP$  Have Zero-Knowledge Proof Systems* Journal of the ACM, vol. 38, n. 1, 1991, pp. 691–729.
- [31] O. Goldreich and Y. Oren, *Definitions and Properties of Zero-Knowledge Proof Systems*, J. of Cryptology, v. 7, n. 1 pp. 1–32, 1994.
- [32] O. Goldreich, R. Ostrovsky, and E. Petrank, *Computational Complexity and Knowledge Complexity*, in Proceedings of the 26th Annual ACM Symposium on Theory of Computing, 1994.
- [33] O. Goldreich and E. Petrank, *Quantifying Knowledge Complexity*, in Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science, 1991.
- [34] S. Goldwasser, S. Micali, and C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems*, SIAM Journal on Computing, vol. 18, n. 1, February 1989.
- [35] R. Impagliazzo and M. Yung, *Direct Minimum Knowledge Computations*, in Proc. of CRYPTO 87.
- [36] M. Ito, A. Saito, and T. Nishizeki, *Secret Sharing Schemes Realizing General Access Structures*, in Proc. of IEEE Globcom 87.
- [37] C. Lund, L. Fortnow, H. Karloff and N. Nisan, *Algebraic Methods for Interactive Proof Systems* in Proceedings of the 31st IEEE Symposium on Foundations of Computer Science, 1990, pp. 2–10.
- [38] M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung, *Perfectly-Secure Zero-Knowledge Arguments Can be Based on General Complexity Assumptions*, in Proc. of CRYPTO 92.
- [39] R. Ostrovsky, R. Venkatesan, and M. Yung, *Interactive Hashing Simplifies Zero-Knowledge Protocol Design*, in Proc. of EUROCRYPT '93.
- [40] E. Petrank, personal communication, Sept. 93.
- [41] A. Shamir, *How to Share a Secret*, Communication of the ACM, vol. 22, n. 11, November 1979, pp. 612–613.
- [42] A. Shamir,  *$IP=PSPACE$* , in Proceedings of the 31st IEEE Symposium on Foundations of Computer Science, 1990, pp. 11–15. (also JACM).
- [43] M. Tompa and H. Woll, *Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information*, in Proceedings of the 28th IEEE Symposium on Foundations of Computer Science, 1987, pp. 472–482.
- [44] L. Valiant, *Short Monotone Formulae for the Majority Function*, Journal of Algorithms, vol. 5, 1984, pp. 363–366.