Keeping the SZK-Verifier Honest Unconditionally

Giovanni Di Crescenzo* Tatsuaki Okamoto** Moti Yung***

Abstract. This paper shows that using direct properties of a zero-knowledge protocol itself, one may impose a honest behavior on the verifier (without additional cryptographic tools). The main technical contribution is showing that if a language L has an Arthur-Merlin (i.e. public coins) honest-verifier statistical SZK proof system then L has an (any-verifier) SZK proof system when we use a non-uniform simulation model of SZK (where the simulation view and protocol view can be made statistically closer than any given polynomial given as a parameter). Three basic questions regarding statistical zero-knowledge (SZK) are solved in this model:

- If L has a honest-verifier SZK proof then L has an any-verifier non-uniform simulation SZK proof.
- If L has an SZK proof then $\overline{L}$ has an non-uniform simulation SZK proof.
- If L has a private-coin SZK proof then L has a public-coin non-uniform simulation SZK proof.

1 Introduction

Statistical zero-knowledge proofs (SZK), introduced by Goldwasser, Micali and Rackoff [13], are an important notion with practical as well as theoretical relevance. In practice, SZK proofs have proved very useful in the design of cryptographic protocols, such as identification schemes [8]. From a theoretical point of view, SZK proofs seem to capture the intrinsic properties of the zero-knowledge concept, since they do not need further cryptographic assumptions, as it is the case for computational zero-knowledge (CZK) proofs. For CZK, all languages in NP [11] and in IP (=PSPACE) [17] (also [4]) are known to have a CZK proof system, while a precise characterization for the languages having SZK proof systems is not known. It is known that the class SZK is in AM $\cap$ co-AM [9, 1], and that NP-complete languages do not have such proofs unless the polynomial hierarchy collapses. Nevertheless, very few properties of SZK have been proved and for many years the problem of establishing unconditional relations among, and properties of SZK proofs, has attracted much attention (see, e.g., [2, 3, 7, 20] and the results below).

* Computer Science and Engineering Dep., University of California San Diego, La Jolla, CA, 92093-0114. E-mail: giovanni@cs.ucsd.edu

** NTT Laboratories, Nippon Telegraph and Telephone Corporation, 1-1 Hikarinooka, Yokosuka-shi, Kanagawa-ken, 239 Japan. E-mail: okamoto@sucaba.isl.ntt.co.jp

*** CertCo, New York NY, E-mail: moti@certco.com, moti@cs.columbia.edu

The notion of zero-knowledge achieved: non-uniform simulation. A few notions of zero-knowledge have been defined in the literature (see [13, 12, 10]). One notion, called *auxiliary-input* zero-knowledge, requires security with respect to any polynomial-time adversary, having an additional auxiliary-input, which models information obtained by the verifier in his past history (which was not required originally). Most protocols in the literature achieve a stronger notion, called *black-box simulation* zero-knowledge, where a simulator treats the verifier as a black box. The simulator may be characterized by an additional parameter, e.g. a polynomial (or a constant). In [10] a black-box simulation was considered where the additional parameter quantifies the random bits used by the simulator. We employ a simulator which uses a sampling technique to assess the bias of the verifier, thus the simulator gets a parameter indicating the sampling bias (which can be made non uniformly smaller than any given polynomial and relaxes the simulation notion which makes the simulator's view smaller than all inverse polynomials). The technique builds a simulation based on simulation of an underlying honest-verifier protocol and preserves the "black-box" property. We call this model, used throughout, *non-uniform (black-box) simulation* statistical zero-knowledge.

Public-coin honest-verifier vs. any-verifier. We show that for any honest-verifier public-coin SZK proof system for a language L there exists an "any-verifier" non-uniform simulation SZK proof system for L. The first unconditional construction for this result was given in [2] and worked for random self-reducible languages. Another such result good only for constant-round proof systems was given in [5] (based on techniques in [19, 22]). Later, two transformations were shown in [6]: one unconditional, for constant-round proof systems, which improved the round-complexity of the transformation in [5], the other for unbounded-round proof systems, assuming one-way functions.

Honest-verifier vs. any-verifier statistical zero-knowledge. Combining our theorem with results in [20], we show a transformation between a proof system which is non-uniform simulation SZK wrt the honest-verifier into one which is SZK wrt any verifier for the same language. This problem was first posed by [2] who solved it under the intractability of discrete log. Later, this problem has been solved in [22], assuming one-way permutations, and, recently, in [20], assuming one-way functions.

SZK for the complemented language. Combining our theorem with results in [20], we show a transformation between a proof system which is SZK for L into one which is non-uniform simulation SZK for the complemented language $\overline{L}$. The first result along these lines was the one in [9], who constructed a (non zero-knowledge) proof system for $\overline{L}$, assuming an SZK proof system for L (a full proof of this was given in [1]). The result then followed from [17, 4] for the case of CZK, assuming one-way functions. Later, this problem was solved by [2] in the case of SZK, assuming the intractability of discrete log. Recently, in [20], an SZK proof system for $\overline{L}$ was given both assuming one-way functions and in the honest-verifier case.

Private-coin vs. public-coin statistical zero-knowledge. Combining our

theorem with results in [20], we show a transformation between a proof system which is private-coin SZK into one which is public-coin non-uniform simulation SZK for the same language. The first result along these lines is due to [14], who proved this transformation between interactive proof systems (i.e., not zero-knowledge). The result then followed from [17, 4] for the case of CZK, assuming one-way functions. Recently, in [20], the result is shown both assuming one-way function, and in the honest-verifier case, for SZK.

2 Non-uniform simulation SZK proof systems

Interactive protocols. Let a pair (A,B) denote an interactive protocol between two interacting probabilistic machines A and B [13]. We denote by x an input common to A and B, by R the content of B's random tape and by y B's auxiliary-input (if any). The *transcript* of an execution of protocol (A,B) on input x, denoted by $\mathrm{tr}_{(A,B)}(x)$, is the messages written by A and B during such execution. By $Out_B(\mathrm{tr}_{(A,B)}(x)) \in \{\mathrm{accept}, \mathrm{reject}\}$ we denote B's *output* at the end of the execution of protocol (A,B) on common input x. We define $View_{B(y)}(x)$, B's view of the interaction with A on input x, as the probability space that assigns to pairs $(R; \mathrm{tr}_{(A,B(y;R))}(x))$ the probability that R is the content of B's random tape and that $\mathrm{tr}_{(A,B(y;R))}(x)$ is the transcript of an execution of protocol (A,B) on common input x given that R is B's random tape and y is B's auxiliary input.

Zero-knowledge proof systems. A zero-knowledge proof system of membership in L is an interactive protocol where the prover convinces a poly-bounded verifier that $x \in L$, without giving additional computational advantage; formally:

Definition 1. *Let P be a probabilistic interactive Turing machine and V a probabilistic poly-time interactive Turing machine sharing input x. Let L be a language. (P,V) is a* SZK PROOF SYSTEM *for L if*

1. (Completeness) For all $x \in L$, $|x| = n$, for all sufficiently large n, and all constants c, $\mathrm{Prob}(Out_V(\mathrm{tr}_{(P,V)}(x)) = accept) = 1 - |x|^c$.

2. (Soundness) For any machine P', for all $x \notin L$, $|x| = n$, for all sufficiently large n, and all constants c, $\mathrm{Prob}(Out_V(\mathrm{tr}_{(P,V)}(x)) = accept) \leq |x|^{-c}$.

3. (Non-uniform Simulation Statistical Zero-Knowledge)
For any probabilistic polynomial-time Turing machine V' There exists a probabilistic Turing machine $S_{V'}$ such that for all $x \in L$, any auxiliary-input y (of size polynomial in $|x|$), and any constant c, it holds that

- *$S_{V'}$ runs in expected polynomial time (which may depend on c);*
- *$\sum_\alpha |\Pr(View_{V'(y)}(x) = \alpha) - \Pr(S_{V'(y)}(x) = \alpha)| \leq |x|^{-c}$.*

We notice that in our definition, the running time of the simulator is expected polynomial time, and the statistical difference between the two spaces is smaller than the inverse of any polynomial, as in the usual definition of black-box simulation SZK. However, the running time of the simulator and the statistical

difference between the two spaces depends also on the constant c (it is true for any constant so it can be made arbitrarily hard to distinguish differences). We will call this relaxed notion, achieving a non-uniform statistical bias based on an input, *non-uniform simulation SZK.*

3 Auxiliary-input cryptographic primitives

In this section we present definitions for auxiliary-input cryptographic primitives with point-wise security arguments (unlike the usual eventually secure notion). They will include distributionally one-way functions, one-way functions, pseudo-random generators, and bit-commitment schemes.

Auxiliary-input one-way functions. Let n be an integer, $aux \in \{0,1\}^n$ be a string, and $f_{aux} : \{0,1\}^n \to \{0,1\}^n$ be an auxiliary-input function. Also, let D_n be a distribution over $\{0,1\}^n$, and U_n be the uniform distribution over $\{0,1\}^n$. We formalize the notion of (locally) breaking and distributionally-breaking an auxiliary-input function.

Definition 2. *We say that an algorithm A_{aux} $(t(n)\epsilon(n))$-breaks the auxiliary-input function f_{aux} on point x if*

$$\text{prob}(y = f_{aux}(x); x' \leftarrow A_{aux}(y) \wedge f_{aux}(x') = y\,) = \epsilon(n),$$

and A_{aux} runs in time $t(n)$ on input 1^n, where the probability is taken over the random coins used by A_{aux}.

Definition 3. *We say that an algorithm A_{aux} $(t(n), \epsilon(n))$-distributionally-breaks the auxiliary-input function f_{aux} on x if*

$$\sum_{\alpha} |\, prob(\, A_{aux}(f_{aux}(x)) \circ f_{aux}(x) = \alpha\,) - prob(\, x \circ f_{aux}(x) = \alpha\,)\,| = \epsilon(n),$$

where by $a \circ b$ we denote concatenation of strings a, b, A_{aux} runs in time $t(n)$ on input 1^n, and the probability is taken over any random coins used by A_{aux}.

Auxiliary-input pseudo-random generators. Let n be an integer, $aux \in \{0,1\}^n$ be a string, and $g_{aux} : \{0,1\}^n \to \{0,1\}^m$, for $m > n$, be an auxiliary-input generator. We formalize the notion of breaking an auxiliary-input generator at a point x.

Definition 4. *We say that algorithm A_{aux} $(t(n), \epsilon(n))$-breaks generator g_{aux} if*

$$\text{prob}(y_1 = g_{aux}(x); y_2 \leftarrow U_{l(n)} : A_{aux}(1^n, y_1) \oplus A_{aux}(1^n, y_2) = 1\,) = \epsilon(n),$$

and A_{aux} runs in time $t(n)$ on input 1^n, where the probability is taken over any random coins used by A_{aux}.

Auxiliary-input bit-commitment schemes.

Definition 5. *Let a pair of interacting Turing machines be Alice which can be an infinitely powerful machine and Bob which is poly-time bounded. An* auxiliary-input bit-commitment scheme $\{(Alice_{aux}, Bob_{aux}(t(n)\epsilon(n)))\}$ *for Alice's input x of size n, is a two-phase interactive protocols (commit and reveal):*

- *After the commit phase, any probabilistic polynomial time $Bob'_{aux}(t(n), \epsilon(n))$ which works $t(n)$ time can compute the bit committed by $Alice_{aux}$ only with probability $\leq 1/2 + \epsilon(n)$.*
- *In the reveal phase, $Alice_{aux}$ can reveal one value (by releasing x). For any $Alice'_{aux}$, if it tries to reveal a bit different from the one committed in the first phase, then Bob_{aux} rejects with overwhelming probability.*

3.1 Extensions to the auxiliary-input case

The following lemmas are simple adaptations of [21, 23, 16, 15, 18] to the auxiliary-input case. They employ reductions between the primitives that relates breaking an input in a given time with a certain probability one primitive to breaking another primitive on a related input with polynomially related time and probability. Lemma 6, due to [21], relates SZK proofs to distributionally one-way functions. It shows that for an input which does not give a distributionally one-way function, the prover's function can be performed efficiently.

Lemma 6. [21] *Let L be a language, (A,B) a (honest-verifier) SZK proof system for L where M is its simulator. Let $x \in \{0,1\}^n$; for all sufficiently large n, there exists an auxiliary-input distributionally one-way function f_{aux} such that: If there exists an algorithm A which, on input $x, pref_i$, $(t(n), \epsilon(n))$-distributionally-breaks f_{aux}, where $aux = x$, then there exists an algorithm A' which, on input x of size n, runs in time $t(n) \cdot poly(n)$ and with overwhelming probability computes r such that $M(r, x) = (pref \circ suff)$, and $B(x, pref \circ suff) = 1$, where* pref *and* suff *are prefix and suffix of M's output, respectively.*

Lemma 7 gives an auxiliary-input one-way function evaluated at a point assuming the existence of an auxiliary-input distributionally one-way function. It follows from a result by Impagliazzo and Luby.

Lemma 7. [16] *Let $aux \in \{0,1\}^n$ and let $\{DF_{aux}\}$ be an auxiliary-input distributionally one-way functions. Then there exists an auxiliary-input one-way functions $\{F_{aux}\}$ such that the following holds. For all sufficiently large n, if there exists an algorithm A_{aux} which $(t(n), \epsilon(n))$-breaks F_{aux} on input x of size n, then there exists an algorithm B_{aux} which $(t'(n), \epsilon'(n))$-distributionally-breaks DF_{aux} on x, where $t'(n) = poly(t(n))$ and $\epsilon'(n) = poly(\epsilon(n))$.*

Lemma 8 gives an auxiliary-input pseudo-random generators assuming the existence of an auxiliary-input one-way function with pointwise translation of hardness. It follows from a result by Håstad, Impagliazzo, Levin and Luby.

Lemma 8. [15] *Let $aux \in \{0,1\}^n$ and let $\{F_{aux}\}$ be an auxiliary-input one-way functions. Then there exists an auxiliary-input pseudo-random generators $\{g_{aux}\}$ such that the following holds. For all sufficiently large n, if there exists an algorithm A_{aux} which $(t(n), \epsilon(n))$-breaks g_{aux} on input x of size n, then there exists an algorithm B_{aux} which $(t'(n), \epsilon'(n))$-breaks F_{aux} on x, where $t'(n) = poly(t(poly(n)))$ and $\epsilon'(n) = poly(\epsilon(poly(n)))$.*

Lemma 9 gives an auxiliary-input bit-commitment schemes assuming the existence of an auxiliary-input pseudo-random generators, which maintains pointwise hardness up to a given polynomial. It follows from a result by Naor.

Lemma 9. [18] *Let $aux \in \{0,1\}^n$ and let $\{g_{aux}\}$ be an auxiliary-input pseudo-random generators. Then there exists an auxiliary-input bit commitment scheme $BC_{aux} = (Alice_{aux}, Bob_{aux})$ such that for all sufficiently large n for Alice's input x of size n, the following two conditions hold:*

1. *If there exists an algorithm Bob'_{aux} which guesses in time $t(n)$ and probability $\epsilon(n)$ the bit committed by $Alice_{aux}$ in the first phase of $(Alice_{aux}, Bob_{aux})$ then there exists an algorithm D_{aux} which $(t'(n), \epsilon'(n))$-breaks g_{aux} on input x, where $t'(n) = poly(t(n))$ and $\epsilon'(n) = poly(\epsilon(n))$.*
2. *The probability that there exists an algorithm $Alice'_{aux}$ which in the second phase reveals a bit different from the one committed in the first phase is negligible.*

Let us try to summarize the above lemmas. Assuming that L has a honest-verifier SZK proofs, Lemma 6 constructs a collection of auxiliary-input distributionally one-way functions. The sequence of Lemmas 7, 8, 9 transforms any auxiliary-input distributionally one-way function into an auxiliary-input one-way functions, which is in turn transformed into an auxiliary-input pseudo-random generators, which in turn transformed into an auxiliary-input strong-to-weak bit-commitment schemes. The hardness of the various primitive is polynomially related locally (from input to one to the same input of the other). Moreover, combining Lemma 6 and Lemma 7, we have that an algorithm which pointwise guesses (given some resources) a bit committed by BC_x can be used to compute an almost uniformly distributed preimage for the prefix of the output of simulator M for (A,B) (for related polynomial resources). We obtain:

Theorem 10. [21, 16, 15, 18] *Let L be a language, let (A,B) be a (honest-verifier) SZK proof system for L. For all sufficiently large n, let $aux \in \{0,1\}^n$. there exists an auxiliary-input commitment schemes $BC_{aux} = (Alice_{aux}, Bob_{aux})$ such that the following two conditions hold:*

1. *If there exists an algorithm Bob'_{aux} which guesses in time $t(n)$ and with probability $\epsilon(n)$ the bit committed by $Alice_{aux}$ in the commit phase of $(Alice_{aux}, Bob_{aux})$ where x of size n is Alice's input, then there exists an algorithm D_{aux} which, on input a prefix $pref_i$ of the output of simulator M ($pref_i$ is a related function of the transcript of the first phase of the*

commitment scheme), runs in time $(poly(t(poly(n))))$ *and with probability* $poly(\epsilon(poly(n)))$ *computes* r *such that* $M(r,x) = (pref_i \circ suff_i)$, *and* $B(x, pref_i \circ suff_i) = 1$. *Moreover, the distribution of string* r *is statistically close to the uniform distribution over all strings* r *such that* $M(r,x) = (pref_i \circ suff_i)$ *and* $B(x, pref_i \circ suff_i) = 1$

2. *The probability that there exists an algorithm* $Alice'_{aux}$ *which in the second phase reveals a bit different from the one committed in the commit phase is negligible.*

4 The transformation

In this section we will prove the following result.

Theorem 11. *Let* L *be a language, let (A,B) be a public-coin honest-verifier SZK proof system for* L, *and let (P,V) be the protocol constructed in Section 4.1. Then (P,V) is an any-verifier non-uniform simulation SZK proof system for* L.

4.1 The protocol (P,V)

Let L be a language and let (A,B) be a public-coin honest-verifier SZK proof system for L. A first step of our transformation is to construct a protocol (C,D) as the parallel execution of n copies of (A,B). Clearly, (C,D) is public-coin and honest-verifier SZK, and we call M the associated simulator (which is the parallel execution of the simulator for (A,B)). We denote by $c(n)$ be the maximum (polynomial) length of a conversation of (C,D), and by $s(n)$ the length of the random string used by M on inputs of size n. In order to construct protocol (P,V), we will construct two functions DF_x, F_x, a generator G_x and a bit-commitment scheme BC_x, where string x is the common input to protocol (P,V). At a very high level, protocol (P,V) can be considered as a way of compiling protocol (C,D) using the bit commitment scheme BC_x where x is the common input. In fact, at any time, protocol (P,V) will use a (single) accepting conversation of protocol (C,D); which we will call the *inner conversation* for protocol (P,V). At the end of the protocol (P,V), V will verify that the inner conversation sent by P is accepting for D. The definition of scheme BC_x, and, more precisely, the definitions of functions F_x and DF_x, will depend on the prefix so far obtained of the inner conversation.

The function F_x. We will define a function F_x in two steps. Informally, in the first step, we would like to construct a function DF_x which, given as input a $s(n)$-bit string R and an index $i \in \{0,1\}^{c(n)}$, returns an i-bit long prefix of the output of the simulator M on input (R,x) concatenated to some padding string $ps = 1 \circ 0^{c(n)-i-1}$. Formally, $DF_x : \{0,1\}^{s(n)} \times \{0,1\}^{\lceil \log c(n) \rceil} \rightarrow \{0,1\}^{c(n)}$ is defined as $DF_x(r,i) = (M(r,x)_{1,\ldots,i}) \circ ps$, where $a_{1,\ldots,i}$ denotes the first i bits of string a. By now, we may think of DF_x as a distributionally one-way function.

Now, we would like to transform function DF_x into a one-way function using the transformation from distributionally one-way functions to one-way functions given in [16]. Precisely, we obtain function F_x by applying a slightly modified

version of this transformation, as follows. Let us consider the mentioned transformation from [16]: it takes a function f as input and returns another function g; now, observe that function g requires multiple independent applications of f. Formally, we define function F_x as the function obtained by applying this transformation to function DF_x, with the exception that one of the applications of DF_x (precisely, a uniformly chosen one) is replaced with the input-output pair $((r,i);(pref \circ ps))$, where $pref$ is the prefix so far in the inner conversation for protocol (P,V), $i = |pref|$, ps is some padding string, and r is uniformly distributed among the strings r such that $\mathrm{M}(r,x) = (x, pref \circ suff)$ and $\mathrm{D}(x, pref \circ suff) = 1$. Since the inner conversation is defined for protocol (C,D), a parallel repetition of (A,B), we will still be able to use the result in [16] for F_x.

The bit commitment scheme $\mathbf{BC}_x$. In order to construct a generator G_x and a bit commitment scheme BC_x, we adapt to the auxiliary-input case results from [15], [18]. Namely, let G_x be the generator that can be constructed starting by the given values of function F_x using the transformation from a one-way function to a pseudo-random generator given in [15]. Finally, let BC_x be the generator that can be constructed starting by generator G_x using the transformation from a pseudo-random generator to a bit-commitment scheme given in [18].

Constructing protocol (P,V). Starting from the assumed public-coin protocol (A,B) for language L, and using scheme BC_x, we construct a protocol (P,V) and show that it is any-verifier SZK.

Previous approaches. The main difficulty with proving that (A,B) is any-verifier SZK is that a dishonest B′ might send messages having a distribution quite different from the uniform one. Given a bit commitment scheme, this problem can be overcome using an idea of [2]. That is, by compiling each step in (A,B) in which B uniformly chooses a bit b and sends it to A, with the following flipping coin protocol. First P commits to a random bit a; then V replies with some possibly biased bit c, and finally the resulting bit b is set equal to $a \oplus c$. Clearly, no matter how V behaves, if P behaves honestly the distribution of bit $a \oplus c$ will be uniform. Now, an idea in [6] is to implement the bit commitment as follows: first, assume that the language L is not in AVBPP (see [23] for definitions), then use the assumption that L has a honest-verifier SZK proof, and sequentially apply known results in the literature to obtain a bit-commitment scheme. Specifically, the results in [21, 23, 15, 18] give a bit commitment scheme.

Our approach. In our protocol we do not make any assumption on the language L. Instead, we adapt results in [21, 24, 16, 15, 18] to the auxiliary-input setting and construct the following auxiliary-input primitives: distributionally one-way function, one-way function, pseudo-random generator and a bit commitment scheme, respectively. One implication of using auxiliary-input primitives in the context of zero-knowledge proofs is that there may exist some x's for which the constructed primitives are not secure. This case considerably complicates the proof of the zero-knowledge property of our protocol. Roughly speaking, we can think of some 'hard' x's for which all primitives are secure, and some 'easy' x's for which they are not, for various level of 'easiness'. In particular, the commitment scheme performed by P seems meaningless in the case of the 'easy'

x's, since it becomes easy for V′ to compute the committed bit. We will overcome this problem with the following strategy where the simulator assesses whether the verifier is cheating or not! More precisely, it will make a close estimate of the probability that the verifier V′ sends a certain random bit, given that he has received a commitment to a certain bit. By looking at this probability, the simulator will be able to compute if the dishonest V′, influences the outcome of the flipping-coin protocol depending on the committed bit (for a given level of influence, where non-uniformity of bias determined by the simulator's input is used for the sampling procedure to work). If the bias of V′ is greater than the given polynomial in n, then the simulator will use V′ as a black box to break the commitment scheme, invert the "one-way" function, and, finally, run the program of the prover. Here, the fact that the construction of the function uses a prefix of the inner conversation so far will help in keeping the following invariant: the simulator always knows a random string that generates the current inner conversation, no matter what is the cheating behavior of the verifier. On the other hand, if the dishonest V′ does not influence the outcome of the flipping-coin protocol (i.e., its level of influence is unnoticeable), then the output bit will be "close enough" to the uniform distribution (it will non-uniformly be smaller than any given polynomial), and therefore the simulator can simulate an execution of the flipping-coin protocol using the usual rewinding technique. Note that the simulation strategy employs the verifier as a black box and thus maintains "black-box simulation". Now, we give more details.

A more formal description of (P,V). Let x, $|x| = n$, be the common input to (P,V). Recall that (A,B) is a public-coin honest-verifier SZK for L, and (C,D) is the parallel execution of n copies of (A,B), and M is the simulator associated with (C,D). Moreover, we assume wlog that the first message in (C,D) is sent by D, and that (C,D) has $r(n)$ rounds; also, each message sent by D has length $k(n)$, and the simulator M uses a random string of length $s(n)$.

The Protocol (P,V)

1. For $i = 1, \ldots, r(n)$,
   P and V set $pref_{i-1} = (a_1, \mathbf{d}_1, \ldots, a_{i-1}, \mathbf{d}_{i-1})$;
   P computes a string $r_i \in \{0,1\}^{s(n)}$ such that $\mathrm{M}(r_i, x) = (pref_{i-1}, a_i, suff)$,
   for some a_i and $\mathrm{B}(x, pref_{i-1}, a_i, suff) = 1$;
   P sends a_i to V;
   for $j = 1, \ldots, k(n)$,
   P uniformly chooses $a_{i,j} \in \{0,1\}$ and a seed $s_{i,j} \in \{0,1\}^{g(n)}$;
   P and V run the commit phase of BC_x,
   where P uses bit $a_{i,j}$ and seed $s_{i,j}$ as his private input;
   V uniformly chooses $c_{i,j} \in \{0,1\}$ and sends it to P;
   P and V run the reveal phase of BC_x and set $d_{i,j} = a_{i,j} \oplus c_{i,j}$;
   P and V set $\mathbf{d}_i = d_{i,1} \circ \cdots \circ d_{i,k(n)}$;
2. V accepts if all verifications are satisfied and
   if D accepts on input transcript $(x, pref_{r(n)-1}, a_{r(n)})$.

4.2 Proof of correctness for (P,V)

This subsection is devoted to show the following

Lemma 12. *If (A,B) is a public-coin honest-verifier SZK for L, then (P,V) is any-verifier non-uniform simulation SZK for L.*

Completeness. Clearly V runs in polynomial time since so do D and the receiver of scheme BC_x. Now, assume $x \in L$, and P,V are honest. In order to show that P can perform his program with high probability it will be enough to show that he can find a string $r_i \in \{0,1\}^{r(n)}$ satisfying $\mathrm{M}(r_i, x) = (x, pref_{i-1}, a_i, suff)$, for some a_i such that transcript $(x, pref_{i-1}, a_i, suff)$ is accepting for (C,D). We observe that since (A,B) is honest-verifier SZK, so is (C,D), and by definition of simulator M, P can always find such a string r_i. Finally, we observe that given that P can successfully perform his program, then the acceptance probability of (P,V) is at least as in (A,B), which is overwhelming.

Soundness. Assume that $x \notin L$. Then notice that the bit $d_{i,j}$ resulting from the output of the j-th execution of the flipping coin protocol compiling round i of (C,D) plays the same role of bit $b_{i,j}$ in the i-th message from D in protocol (C,D). Now, we show that for any P*, bit $d_{i,j}$ is almost uniformly distributed. To see this, notice that bit $d_{i,j}$ is computed as $c_{i,j} \oplus a_{i,j}$, where bit $c_{i,j}$ is uniformly chosen by V and bit $a_{i,j}$ is the bit decommitted by P', using the scheme in [18]. Now, from property 2 of Lemma 9, we obtain that for any x, P' can decommit two possible values for bit $a_{i,j}$ only with negligible probability, for any x (both in L or not, when not in L the commitment may be non-concealing– but we do not care). Following the analysis done in Claim 3.1 of [18], one can show that the probability that there exist two seeds $s_{i,j}, s'_{i,j}$ that can be used as decommitments of two distinct bits is at most $2^{-g(n)}$, which is negligible. This implies that the distribution of bit $a_{i,j}$ is almost uniform; namely, the probability that $a_{i,j} = b$ is different from $1/2$ only by at most a negligible factor, for $b = 0, 1$, and the same holds for bit $d_{i,j}$. Now, since there are at most a polynomial number of bits $d_{i,j}$, the probability that all bits $d_{i,j}$ are not independently and uniformly distributed is negligible. Thus, with probability 1– a negligible factor, bits $d_{i,j}$ are distributed exactly as bits $b_{i,j}$ in (C,D). This implies that, if $x \notin L$, and for any P', the probability that V accepts is equal to a negligible factor plus the error probability in the soundness of (C,D), which is negligible.

Any-verifier non-uniform simulation statistical zero-knowledge. We show that for (P,V) for any probabilistic polynomial-time verifier V', there exists a simulator $S_{V'}$, such that, for any $x \in L$, and for any constant c, the statistical distance between $\mathrm{S}_{V'}(x)$ and $\mathrm{View}_{V'}(x)$ is at most $|x|^{-c}$. The simulation is black-box whenever the one of (A,B) is. Informally, our simulator S will try to simulate an accepting conversation of (P,V), as follows. First, S will generate a conversation of (C,D), and then will try to force the outcomes of the flipping-coin subprotocols executed by (P,V) consistently with the messages of the verifier D, in the inner conversation. In this process, the simulator S will use the rewinding simulation technique to obtain the desired outcome of any flipping-coin protocol. Contrarily to what usually happens, this strategy may not be successful in this case, especially if the cheating verifier V' somehow is able to guess the bit committed by P. We will show that such a cheating verifier can be used as a

black box to break the commitment scheme BC_x. Now, notice that by Lemma 10 an algorithm breaking scheme BC_x can be transformed into an algorithm that inverts a prefix of the output of the simulator M for (C,D). The construction of our proof system (P,V) will guarantee that the preimage thus obtained has distance smaller than any given (input) polynomial from the uniform distribution among those strings generating the given prefix of the inner conversation. This will allow us to generate a random string for the simulator which generates the inner conversation obtained so far. Thus, for any cheating verifier V', the simulation will always make some progress, either thanks to the rewinding technique, or because the cheating behavior of V' allows us to compute a random string which generates the inner conversation so far. In this way, $r(n)k(n)$ phases are sufficient to guarantee that the simulator outputs an accepting conversation of protocol (P,V). Now we proceed more formally.

The procedures Estimate, Guess and Invert. The algorithm S will use three procedures. The first procedure, called Estimate will be run by S to evaluate the cheating behavior of V' in each execution of a flipping-coin protocol. Specifically, notice that for some ('easy' or 'hard') x's, V' might be able to correctly guess the committed bit $a_{i,j}$ with some probability bounded away from 1/2, and then influence the distribution of the bit $d_{i,j}$ output of the flipping-coin protocol. Using procedure Estimate for each execution of the flipping-coin protocol, the simulator will estimate the bias $\delta_{i,j}$ caused by V' on bit $d_{i,j}$. The estimate will be done as follows: on input a constant c let $r(n) = n^{r'}, k(n) = n^{k'}$, and let $q = \max\{100, 2(c+r'k')+1\}$, S will try to estimate $d_{i,j}$ with a number $e_{i,j}$ such that $e_{i,j}$ is near 1/2: $e_{i,j} \in [1/2 - n^{-q}, 1/2 + n^{-q}]$ (i.e., be unnoticeable) with exponentially small probability of errors, or not (i.e., $e_{i,j}$ is bounded away from 1/2, e.g. $e_{i,j} \in [0, 1/2-3n^{-q}] \cup [1/2+3n^{-q}, 1]$) Using a large enough polynomial-size sample n^s, extreme assumption on the bias probability, and Chernoff bounds, an exponentially small error in the estimate is possible (using an appropriate ϵ in the estimation test below). Thus the accuracy error of this estimate will be less than n^{-q}, which is enough for us since we would like the distance between the output of the simulator and the output of the protocol to be smaller than n^{-c}.

Now, on input the common input to the proof system x, the constant q, the number i of round and the number j of bit in the i-th round of (C,D), procedure Estimate runs the following steps:

Procedure Estimate:

1. set $count_{0,0} = count_{0,1} = count_{1,0} = count_{1,1} = 0$;
2. for $l = 1, \ldots, n^s$,
   uniformly choose bit a_l and commit to a_l using scheme BC_x;
   get bit c_l from V';
   set $count_{a_l,c_l} = count_{a_l,c_l} + 1$;
   reveal bit a_l to V' using scheme BC_x;
   rewind V' to the state just before running the j-th execution of protocol BC_x in the i-th round;
3. set $p_{h,k} = count_{h,k}/n^{2q+1}$, for $h, k = 0, 1$, and $\delta_{i,j} = |p_{0,1} + p_{1,0} - p_{1,1} - p_{0,0}|$;

4. if $\delta_{i,j} > \epsilon$ then set $bias = yes$ else set $bias = no$;
5. output $(bias, p_{0,0}, p_{1,0}, p_{0,1}, p_{1,1})$ and return.

Procedure Estimate will be used by the simulator $S_{V'}$ to distinguish whether the bias added by the verifier V' to the distribution of the output of a flipping coin protocol is at least n^{-q} or not. In fact, the procedure satisfies the following.

Fact 13. Let $bias$ be output by procedure Estimate on input x, i, j. The bias $\delta_{i,j}$ of bit $b_{i,j}$ in an execution of (P,V') is at least n^{-q} if $bias = yes$ or smaller than n^{-q} if $bias = no$.

In the case $bias = no$, Fact 13 guarantees that the bias $\delta_{i,j}$ is smaller than n^{-q}. Then the simulator $S_{V'}$ is able to successfully simulate the j-th execution in the i-th round of the bit commitment scheme BC_x, by using the rewinding technique until it holds that $d_{i,j} = b_{i,j}$. This happens in an expected number of steps that is at most $2n^q$. In the second case, namely, when $bias = yes$, the simulator $S_{V'}$ is *not* able to successfully simulate the j-th execution in the i-th round of the bit commitment scheme BC_x. However, since the distribution of bit $b_{i,j}$ is far from uniform, with sufficiently high probability the value of $c_{i,j}$ is chosen by V' depending on that of $a_{i,j}$, and thus V' can break the commitment scheme with high probability. Formally, on input x, and given the transcript $tr_{i,j}$ of the commit phase of an execution of scheme BC_x, procedure Guess does the following:

Procedure Guess:

1. Let $tr_{i,j}$ be the transcript of the commit phase of an execution of BC_x with V';
2. get bit $c_{i,j}$ from V';
3. if $p_{1,1} + p_{0,0} > p_{0,1} + p_{1,0}$ then set $a_{i,j} = c_{i,j}$ else set $a_{i,j} = 1 - c_{i,j}$;
4. output $a_{i,j}$ and return.

Procedure Guess assumed (for exposition) uniform behavior on cheating on commitment of 1 and commitment of 0. A refinement leading to a guess in other behaviors is possible. The procedure satisfies the following fact

Fact 14. The bit $a_{i,j}$ output by procedure Guess on input $x, i, j, tr_{i,j}$ is equal to the bit committed by P in transcript $tr_{i,j}$ with probability at least $\delta_{i,j}$.

From Fact 14 it follows that there exists an algorithm which breaks the commitment scheme BC_x. Then, using the fact that protocol (C,D) is the parallel execution of protocol (A,B), together with the reduction in Theorem 10, we obtain that there exists an algorithm Invert which (using appropriate polynomial resources and success probability), given input $x, pref$, returns a randomly chosen string which allows the simulator M to generate the current prefix of the inner conversation. Formally, we obtain the following

Fact 15. Let r be the output of algorithm Invert on input $x, pref_i$. Then the distribution of r is statistically close to the uniform distribution on the strings r such that $M(r, x) = (x, pref_i \circ suff_i)$, and $D(x, pref_i \circ suff_i) = 1$.

The simulator $S_{V'}$. On input x, and constant c, algorithm $S_{V'}$ uses procedures Estimate and Invert, as follows.

1. Uniformly choose an $s(n)$-bit string R.
2. Run M on input R, x thus obtaining conversation
   $conv = ((b_{1,1}, \ldots, b_{1,tk}), a_1, \ldots, a_{r-1}, (b_{r,1}, \ldots, b_{r,tk}), a_r)$ as output.
3. For $i = 1, \cdots, r$,
   for $j = 1, \ldots, tk$,
   run procedure Estimate on input x, i, j, c, and let *bias* be its output;
   if $bias = no$ then
   repeat
   rewind V′ until after message a_{i-1} was sent to him;
   run a flipping coin protocol interacting with V′
   and let $d_{i,j}$ be the resulting output;
   until $d_{i,j} = b_{i,j}$;
   if $bias = yes$ then
   run a flipping coin protocol interacting with V′
   and let $b_{i,j}$ be the resulting output;
   rewind V′ until after message a_{i-1} was sent to him;
   set $pref_i = (\mathrm{b}_1, a_1, \ldots, \mathrm{b}_{i-1}, a_{i-1}, (b_{i,1}, \ldots, b_{i,j}))$;
   run algorithm Invert on input $x, pref_i$, and let R_i be its output;
   set $conv = (\mathrm{b}_1, a_1, \ldots, a_{r-1}, \mathrm{b}_r, a_r) = M(R_i, x)$;
   send message a_i to V′.
4. Output: *conv* and halt.

We observe that the expected running time of simulator S is $poly(n) \cdot n^{O(q)}$, and thus it is expected polynomial time, for any given q derived from any given constant c. Now we need to show that the output of the simulator is statistically close to the view of the verifier.
In the next lemma we prove that the output of the simulator is statistically close to the view of the verifier.

Lemma 16. *For any common input x and auxiliary input y, for any constant c, the statistical distance between the output of $S_{V'}$ on input x, c and the view of V' in protocol (P, V') on input x is at most $|x|^{-c}$.*

Proof. All messages from V′ are equally distributed in both spaces since they are computed in the same way. Now, we consider the messages from the prover in both spaces. The messages sent by the prover in the commitment phase of BC_x are computed almost in the same way in both spaces; here, the only difference is that the output of each flipping coin protocol is uniformly distributed in the simulation (since it is equal to the random bit output by M) while it has a bias smaller than n^{-q} in the view of V′. However, this contributes a factor smaller than n^{-c} to the statistical difference between the two spaces. Now, we consider the messages a_i from the prover, for $i = 1, \ldots, r$. We observe that in the output of S they are all computed in two ways: either they are taken from the output of M on input the uniformly distributed string R and x, or they are computed by first using procedure Invert to compute a random string R_i for M and then they are taken from the output of M on input R_i and x. In the first case it holds

that in both spaces the message a_i is distributed as a message output from M on a random string. In the second case, the distribution of this message in the output of the protocol is as in the first case. In the output of the simulator S, instead, the procedure Invert is executed. Now, notice that, given prefix $pref_i$ of a conversation, procedure Invert computes a string R_i which, by Fact 15, is chosen uniformly enough among those satisfying $M(R_i, x) = (pref_i, suff_i)$, and $D(x, pref_i, suff_i) = 1$. In other words, message a_{i+1} is computed with a distribution statistically close to that of the simulator $S_{V'}$, conditioned by the conversation so far, which is statistically close to the prover's distribution. Therefore, the overall statistical distance between the two spaces is less than n^{-c}. □

5 Implications on non-uniform simulation SZK proofs

We combine the result in Theorem 11 with some results in the literature. This will show results regarding non-uniform simulation SZK.

Theorem 17. *Let L be a language. If L has a honest-verifier SZK proof then L has a non-uniform sim. SZK proof.*

Proof. Assume L has a honest-verifier SZK proof system (A,B). If (A,B) is public-coin, applying Theorem 11 will prove the result. If (A,B) is private-coin, then a result in [20] allows to obtain a public-coin honest-verifier SZK proof system, and, then, applying again Theorem 11 will prove the result. □

Theorem 18. *Let L be a language. If L has a private-coin SZK proof then L has a non-uniform sim. public-coin SZK proof.*

Proof. Assume L has a private-coin SZK proof system (A,B). Using a result in [20], (A,B) can be transformed into a honest-verifier public-coin SZK proof system (C,D) for L. Then, using Theorem 11, (C,D) can be transformed into an SZK proof system for L. □

Theorem 19. *Let L be a language. If L has a SZK proof then $\overline{L}$ has a non-uniform sim. SZK proof.*

Proof. Assume L has a SZK proof system (A,B). Using a result in [20], (A,B) can be transformed into a honest-verifier SZK proof system (C,) for $\overline{L}$. Then, using Theorem 17, (C,D) can be transformed into a SZK proof system for $\overline{L}$. □

Acknowledgements. Many thanks go to Alfredo De Santis, Oded Goldreich and Russell Impagliazzo for valuable discussions and remarks.

References

1. W. Aiello and J. Håstad. *Statistical Zero Knowledge Can Be Recognized in Two Rounds*, Journal of Computer and System Sciences, vol. 42, 1991, pp. 327–345.
2. M. Bellare, S. Micali, and R. Ostrovsky, *The (True) Complexity of Statistical Zero-Knowledge Proofs*, in STOC 90.
3. M. Bellare, and E. Petrank, *Making Zero-Knowledge Provers Efficient*, STOC 92.
4. M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway, *Everything Provable is Provable in Zero Knowledge*, in CRYPTO 88.
5. I. Damgård, *Interactive Hashing can Simplify Zero-Knowledge Design without Complexity assumptions*, in CRYPTO 92.
6. I. Damgård, O. Goldreich, T. Okamoto, and A. Wigderson, *Honest-Verifier vs. Dishonest-Verifier in Public-Coin Zero-Knowledge Proofs*, in CRYPTO 95.
7. A. De Santis, G. Di Crescenzo, P. Persiano, and M. Yung, *On Monotone Formula Closure of SZK*, in FOCS 94.
8. U. Feige, A. Fiat, and A. Shamir, *Zero-Knowledge Proofs of Identity*, Journal of Cryptology, vol. 1, 1988, pp. 77–94.
9. L. Fortnow, *The Complexity of Perfect Zero Knowledge*, in STOC 87.
10. O. Goldreich and H. Krawczyk, *On the Composition of Zero-Knowledge Proof Systems*, SIAM Journal on Computing, 1996.
11. 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.
12. O. Goldreich and Y. Oren, *Definitions and Properties of Zero-Knowledge Proof Systems*, Journal of Cryptology, v. 7, n. 1, 1994.
13. S. Goldwasser, S. Micali, and C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems*, SIAM Journal on Computing, vol. 18, n. 1, February 1989.
14. S. Goldwasser and M. Sipser, *Private Coins versus Public Coins in Interactive Proof–Systems*, in STOC 1986.
15. J. Håstad, R. Impagliazzo, L. Levin, and M. Luby, *Construction of a Pseudo-Random Generator from One-Way Function*, to appear in SIAM Journal on Computing, previous versions: FOCS 89 and STOC 90.
16. R. Impagliazzo and M. Luby, *One-Way Functions are Necessary for Complexity-Based Cryptography*, in FOCS 90.
17. R. Impagliazzo and M. Yung, *Direct Minimum Knowledge Computations*, in CRYPTO 87.
18. M. Naor, *Bit-Commitment Using Pseudo-Randomness*, in CRYPTO 89.
19. M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung, *Perfectly-Secure Zero-Knowledge Arguments Can be Based on General Complexity Assumptions*, in CRYPTO 92.
20. T. Okamoto, *On Relations Between Statistical Zero-Knowledge Proofs*, STOC 96.
21. R. Ostrovsky, *One-Way Functions, Hard on Average Problems and Statistical Zero-Knowledge Proofs*, in Structures 91.
22. R. Ostrovsky, R. Venkatesan, and M. Yung, *Interactive Hashing Simplifies Zero-Knowledge Protocol Design*, in EUROCRYPT '93.
23. R. Ostrovsky, and A. Wigderson, *One-way Functions are Essential for Non-Trivial Zero-Knowledge*, in ISTCS 93.
24. A. Yao, *Theory and Applications of Trapdoor Functions*, in FOCS 81.