# On Monotone Function Closure of Perfect and Statistical Zero-Knowledge

Extended Abstract

Ivan Damgård, Aarhus University, BRICS
and
Ronald Cramer, CWI

**Abstract**

Assume we are given a language $L$ with an honest verifier perfect zero-knowledge proof system. Assume also that the proof system is a $\leq 3$ move Arthur-Merlin game. The class of such languages includes all random self-reducible language, and also any language with a perfect zero-knowledge non-interactive proof. We show that such a language satisfies a certain closure property, namely that languages constructed from $L$ by applying certain monotone functions to statements on membership in $L$ have perfect zero-knowledge proof systems. The new set of languages we can build includes $L$ itself, but also for example languages consisting of $n$ words of which at least $t \leq n$ are in $L$. A similar closure property is shown to hold for the complement of $L$ and for statistical zero-knowledge. The property we need for the monotone functions used to build the new languages is that there are efficient secret sharing schemes for their associated access structures. This includes (but is not necessarily limited to) all monotone functions with polynomial size monotone formulas.

## 1 Introduction

Zero-knowledge proofs, introduced by Goldwasser, Micali and Rackoff [8], offer the possibility of proving a statement while revealing nothing but the validity of the assertion. Perfect and statistical zero-knowledge proofs are special cases of zero-knowledge for which this property can be proved without resorting to unproven complexity assumptions. Structural properties of the classes of languages with perfect or statistical zero-knowledge proofs

(PZKIP, SZKIP) - such as the ones implied by the results in this paper - are interesting for this reason, but also since the status of PZKIP and SZKIP w.r.t. classical classes is not well understood. It is unlikely that they contain NP, since the polynomial hierachy would collapse if they did [6], on the other hand they are not known to be contained in NP, since problems such as graph non-isomorphism is in PZKIP.

From a practical point of view, we note that extremely efficient perfect zero-knowledge proofs are known. Hence efficient techniques for building new perfect or statistical zero-knowledge proofs from known ones, using for instance the closure properties presented here, have potential applications in practice.

The monotone closure property studied in this paper can be informally described by considering a situation where we are given $n$ *atomic statements*, e.g. statements of the form $x \in L$ for a word $x$ and a language $L$, and a proof system $\mathcal{P}$ by which an atomic statement (if true) can be proved in perfect (or statistical) ZK. If we also have a monotone Boolean function $f$ with $n$ inputs, we can form a new statement in a natural way by letting the validity of each atomic statement correspond to an input bit of $f$. For example, if $f$ is a threshold function with threshold $t$, the new statement would say that at least $t$ of the atoms are true. Informally, we say that monotone closure holds for a proof system $\mathcal{P}$, if $\mathcal{P}$ can be used to construct perfect zero-knowledge proofs for new statements of the form just described.

This idea of using monotone operations to build new interactive proofs from known ones with certain properties was introduced by De Santis et al. in [7] for statistical zero-knowledge proofs and independently by Cramer et al. in [4] for witness hiding proofs of knowledge.

The results in [7] apply to a quite specialized subclass of languages in PZKIP, namely random self-reducible languages (RSR). In this paper, we show that monotone closure applies to a much more general class, namely languages with 3-move AM proofs that are statistical honest verifier ZK. In addition to RSR, this class contains e.g. perfect and statistical non-interactive ZK[1]. Hence it contains for example the language of Blum-integers [9], the language of so called 2-regular integers [3] and the language of functions that are almost permutations [1]. None of these problems seem to be in RSR.

This substantially extends the set of languages known to be in PZKIP

[1]an interactive proof can be obtained by letting the verifier choose the random reference string, send it to the prover and have the prover reply with the (non-interactive) proof

and SZKIP, but we also believe that it provides a better understanding of the essential properties needed for monotone closure: It holds if the given proof system has a certain form, and not because of algebraic properties of the underlying language, such as random self-reducibility.

Another important point is that it might seem from [7] and [4] that monotone closure holds only for protocols that satisfy a very strong and non-standard requirement for soundness, namely that a cheating prover can never answer more than 1 of the verifier's potential questions. We show that this is in fact unnecessary - the standard requirement for soundness suffices.

The key to our results is a combination of the techniques from [7] and [4] with the interactive hashing technique of Naor et al. [11].

## 1.1 Summary of Results

In the following we will say that the language $L$ is in HV3AM/PZKIP, resp. HV3AM/SZKIP if it has an interactive proof system satisfying the following:

1. It is perfect, resp. statistical, honest verifier zero-knowledge.
2. It is an Arthur-Merlin game with at most 3 moves.
3. Completeness is satisfied with probability 1 (i.e. the honest prover always succeeds in convincing the verifier about a true statement).

Such a proof system is called an HV3AM/PZKIP-proof, respectively an HV3AM/SZKIP-proof.

Some of our results can be shown without the third condition above, but we have included it globally for simplicity. We know of no concrete protocol that satisfies the first two conditions, but not the third.

We will need the concept of a *parsing scheme* $\pi$. Informally, this is a polynomial time algorithm which takes a word $x$ as input and divides it into a concatenation of $n$ words $x_1, ..., x_n$. A parsing is admissible, if the length of any $x_i$ is polynomially related to the length of $x$, i.e., the size of any $x_i$ is at least equal to some fixed non-constant polynomial (in $|x|$). This is a technicality which is needed in the following to preserve soundness and statistical ZK, and will be made presice in the next section.

We will assume we are given a family of monotone functions $F = \{f_n | \, n = 1, 2, ..\}$, where $f_n : \{0,1\}^n \rightarrow \{0,1\}$. We will use $F^*$ to denote the family of monotone functions containing the duals[2] of those in $F$.

[2]The dual $f_n^* : \{0,1\}^n \rightarrow \{0,1\}$ of $f_n$ is defined as $f_n^*(x) = 1$ iff $f_n(x \oplus \mathbf{1}) = 0$.

The language $F(\pi, L)$ is constructed as follows from $F, \pi$ and $L$: given a word $x$, parse it using $\pi$ to divide $x$ into $x_1, .., x_n$. Define the bit string $B = b_1, ..., b_n$ by $b_i = 1$ if and only if $x_i \in L$. Then $x \in F(\pi, L)$ if and only if $f_n(B) = 1$.

Note that it is easy to choose $F$ and $\pi$ such that for any $L$, $L = F(\pi, L)$.

A final technical ingredient we need is a secret sharing scheme, which is consistent with the access structures defined by the functions in $F$ or in $F^*$. The reader not familiar with these concepts for now only needs to know that a secret sharing scheme is a probabilistic algorithm that from an input secret will compute a number of shares. The original secret can then be reconstructed, only from certain subsets of the shares. If the secret sharing runs in polynomial time and the distribution of the shares has some specific properties defined in terms of $F$ ($F^*$), we say that $F$ ($F^*$) has an efficient secret sharing scheme with completion. A precise definition can be found in the following section.

We prove the following results:

**Theorem 1.1** *If $L \in HV3AM/PZKIP$, resp. $L \in HV3AM/SZKIP$ and the monotone function family $F^*$ has an efficient secret sharing scheme with completion, then for any admissible parsing $\pi$, $F(\pi, L) \in PZKIP$, resp. $F(\pi, L) \in SZKIP$.*

We remark that Benaloh and Leichter [2] show how to build an efficient secret sharing scheme (which can be shown to satisfy the completion property) for any monotone formula. Hence a trivial consequence of Theorem 1.1 and [2] is

**Corollary 1.2** *If $L \in HV3AM/PZKIP$, resp. $L \in HV3AM/SZKIP$ and functions in the family $F$ have polynomial size monotone formulae then for any admissible parsing $\pi$, $F(\pi, L) \in PZKIP$, resp. $F(\pi, L) \in SZKIP$.*

We do not know if Theorem 1.1 is strictly stronger than Corollary 1.2, but conjecture that the answer is yes. Our assumption that $F^*$ has an efficient secret sharing scheme with completion implies that functions in $F$ can be computed in time polynomial in $k$, but it is known [13] that monotone functions, such as perfect matching in a bipartite graph, exist that can be computed in polynomial time, but require exponential size monotone formulae. However, to the best of our knowledge, no efficient secret sharing schemes are known for the functions studied in [13].

Let $L'$ be the complement of $L$, and let $HVSZKIP$ be the classes of languages with honest verifier statistical zero-knowledge proofs. We show that

**Theorem 1.3** *If $L \in HV3AM/SZKIP$ and the family $F$ has an efficient secret sharing scheme with completion, then for any admissible parsing $\pi$, $F(\pi, L') \in HVSZKIP$.*

**Corollary 1.4** *Suppose $L \in HV3AM/SZKIP$, that computing witnesses for membership in $L$ is hard, that the proof system for $L$ is also a proof of knowledge and the family $F$ has an efficient secret sharing scheme with completion. Then for any admissible parsing $\pi$, $F(\pi, L') \in SZKIP$.*

We remark that the extra assumptions on $L$ in the corollary are only needed to ensure that we have a unconditionally hiding bit commitment scheme, by [5]. Any other assumption that would ensure this could be substituted, e.g. existence of one-way permutations [11] or existence of collision intractable hash functions [12].

All our proofs are constructive and preserve efficiency in the sense that the new provers and verifiers we construct are polynomial time machines that call the prover and verifier of the original proof system for $L$ as subrutines.

Finally, it is interesting to note that our construction starts with a protocol that is only honest verifier zero-knowledge, and produces a protocol that is zero-knowledge in general. Thus our results contain parts of those of [5] as a specical case.

## 2 Notation and Definitions

Consider a monotone function family $F$. In the following, if $A \subset \{1, ..., n\}$, $f_n(A)$ means $f_n$ evaluated on a bit string with 1's in positions corresponding to elements in $A$ and 0's elsewhere.

An *efficient secret sharing scheme with completion* $S$ for $F$ is a probabilistic algorithm that takes a (secret) bit $b_s$ and integer $n$ as input and outputs $n$ shares $s_1, ..., s_n$. $S$ should run in time polynomial in $n$, whence the length of the $i$'th share is upper bounded by a polynomial $l_i(n)$. The following conditions must be satisfied:

1. For any $A$ such that $f_n(A) = 1$, $b_s$ can be computed from $\{s_i | \ i \in A\}$ in time polynomial in $n$, whereas if $f_n(A) = 0$, the distribution of

$\{s_i| \ i \in A\}$ generated from $b_s = 1$ is the same as when $b_s = 0$; i.e. $\{s_i| \ i \in A\}$ gives no information about $b_s$.

2. Given a full set of $n$ shares and a bit $b_s$, it must be possible to test in time polynomial in $n$ that the shares are all consistent with $b_s$, i.e. are a possible result of running $S$ on input $n$ and $b_s$.

3. If $B$ is a set for which $f_n(B) = 0$, then the distribution of the subset of shares corresponding to $B$ must be independent of the secret bit. Call this distribution $D_B$. We require that given a subset of shares distributed according to $D_B$ and a bit $b_s$, it is always possible in polynomial time to complete this subset to a full set of shares consistent with $b_s$ and with the same distribution as the output distribution of $S$.

A *parsing scheme* is a polynomial time algorithm which takes a $k$-bit word $x$ as input and outputs integers $n, i_0, i_1, ..., i_n$, where $0 = i_0 < i_1 < ... < i_{n-1} < i_n = k$. Note that $n$ may depend on $x$. Having run $\pi$ on input $x$ we can view $x$ as a concatenation of words $x = x_1|x_2|...|x_n$, where $x_j$ consists of bits $i_{j-1}+1$ through $i_j$ of $x$. A parsing scheme is *admissible* if there is a fixed non-constant polynomial $q$ such that the length of any $x_i$ is at least $q(k)$.

Next, we define the basic model for the protocols we will consider. The protocol takes place between a prover $P$ and a verifier $V$, where $(P, V)$ is a pair of interactive Turing machines as defined in [8]. The common input to the prover and verifier is usually called $x$. It is chosen from some language $L$, and its length is denoted by $k$. The verifier is assumed to run in probabilistic polymial time in $k$. The contents of the random tape of $P$ is called $R_P$. The contents of the knowledge and random tapes of $V$ are called $K_V$, resp. $R_V$. At the end of the conversation, $V$ outputs "accept" or "reject".

We will start from protocols with at most 3 moves where the verifier sends only independent random bits (an Arthur-Merlin game). This is defined as follows: the prover first sends a message $a = a(x, R_P)$, the verifier sends a random challenge $c$ consisting of $t = t(k)$ random bits, and finally the prover sends a response $r = r(x, R_P, c)$. The verifier now evaluates a predicate $\phi(x, a, c, r)$ on the common input and the conversation and outputs "accept" or "reject" according to the result of this computation. The *view* of participant $X$ ($= P$ or $V$) is defined to be the concatenation of all inputs to $X$ and all messages sent in the protocol.

For definitions of soundness, completeness and perfect/statistical/computational zero-knowledge, we refer to [8]. In particular, soundness will be taken to mean that a cheating prover can only convince the verifier with superpolynomially small probability. In addition, we define $(P, V)$ to be *honest verifier zero-knowledge* (HVZK) if there is an expected polynomial time probabilistic machine $M$, which on input $x \in L$ produces a view that is computationally indistinguishable from the view of $V$ produced on input $x \in L$, with independent random bits on the random tape and a blank knowledge tape. Perfect honest verifier zero-knowledge and statistical honest verifier zero-knowledge are defined similarly, but we require perfect, resp. statistical indistinguishability. We will assume for simplicity that all honest verifier simulators run in expected polynomial time on all inputs, even words not in $L$ (although in this case they may not produce accepting conversations). Our results would remain essentially the same without this assumption.

In the following, a function $p(k) \geq 0$ will be called *negligible in k* if, for any polynomial $f$, it holds that $p(k) \leq 1/f(k)$ for all sufficiently large $k$.

## 3 Interactive Hashing

For our main results, we need the technique from [11] of interactive hashing, which we repeat here with some minor changes to match our context.

The purpose of an interactive hashing game is to allow two mutually distrusting parties to jointly select an element at random from some (large) set. The idea is that in each step one party will cut the remaining set of possibilites in two halves, and the other party will choose which half they should continue with. Since specifying a random cut in a large set requires too many bits, a 2-universal hash function with a one-bit output is used in stead to specify the cut.

To link the hashing game to the protocol model from the last section, we need the following notation: We assume that a predicate $\phi'$ is given that takes as input pairs $(c, r)$, where $c$ is of length $t$ bits, and $r$ has length polynomially related to $t$. If $\phi'(c, r) = 1$, we say that $r$ is *correct with respect to c*. The interactive hashing takes place between parties $P$ and $V$, and we assume that $P$ for any $c$ is capable of computing an $r$ that is correct w.r.t. $c$.

The interactive hashing now goes as follows:

1. $P$ selects a random vector $c \in GF(2)^t$ which is kept secret from $V$. This $c$ is called *$P$'s original choice.*

2. $V$ selects at random $t-1$ linearly independent vectors $h_1, ..., h_{t-1}$ in $GF(2)^t$.

3. Repeat the following for $j = 1..t-1$: $V$ sends $h_j$ to $P$, and $P$ sends $b_j = h_j \cdot c$ (the inner product) to $V$.

4. Both parties compute the two vectors $c_0, c_1$ with the property that $b_j = h_j \cdot c_i$ for all $j = 1..t-1$. $P$ decides in an arbitrary way which of the two vectors is called $c_0$ and announces this to $V$. Of course, $P$'s original choice $c$ is equal to one of $c_0, c_1$.

5. $V$ chooses $v = 0$ or 1 at random and sends it to $P$.

6. $P$ computes an $r$ that is correct w.r.t. $c_v$ and sends it to $V$.

In [11], the following is proved about this procedure:

**Lemma 3.1** *At the end of step 4 above, an arbitrary cheating $V^*$ has no Shannon information about which of $c_0, c_1$ is $P$'s original choice.*

**Lemma 3.2** *Let $P^*$ be any machine playing the role of $P$ in the interactive hashing. Assume that $P^*$ can return correct $r$-values for both $c_0$ and $c_1$ with probability $\epsilon$, taken over the coin tosses of $V$ and $P^*$. There is a probabilistic polynomial time machine $M$ using any such $P^*$ as an oracle that can, on input a random $c \in GF(2)^t$, compute an $r$ correct with respect to $c$ with probability $T(t, \epsilon)$, where $T$ is a function that is polynomial in both $1/t$ and $\epsilon$. The probability is over the choice of $c$ and the coin tosses of $M$.*

### 3.1 Using Interactive Hashing in the Protocol

Suppose now that we have a HV3AM/PZKIP-proof (or HV3AM/SZKIP) $(P_0, V_0)$ for the language $L$. To show our main results, we would like a situation where we could assume for any $x \notin L$, that no matter how the prover selects the first message $a$, at most 1 of the possible challenges from $V_0$ can be answered correctly. Unfortunately, this is not the case in general, even if the protocol satisfies soundness in the usual sense. We can, however, get into the situation we want (at least with large probability), if we let the challenge to be answered be selected using interactive hashing instead of letting the verifier select it on his own. The following protocol $(P_0^{IH}, V_0^{IH})$, derived from $(P_0, V_0)$ using interactive hashing, has this property.

**The protocol** $(P_0^{IH}, V_0^{IH})$

1. $P_0^{IH}$ copies random bits into the random tape of $P_0$. It then starts running $P_0$ on input $x$. This results in a first message $a$. This is sent to $V_0^{IH}$.

2. $P_0^{IH}$ and $V_0^{IH}$ go through the interactive hashing process described above, playing the roles of $P$ and $V$, respectively. The purpose will be to agree on the challenge to be answered. We define the predicate $\phi'$ by $\phi'(c, r) = \phi(x, a, c, r)$, i.e. a correct $r$ is a value that will make $V_0$ accept. At the beginning of step 6 of the interactive hashing, two values $c_0, c_1$ have already been isolated, and $V_0^{IH}$ has sent a bit $v$. $P_0^{IH}$ can now obtain a correct $r$-value by passing $c_v$ to $P_0$ and relaying the $r$ returned to $V_0^{IH}$.

3. $V_0^{IH}$ uses $V_0$ to decide if the conversation $(a, c_v, r)$ would lead to accept. If so, $V_0^{IH}$ outputs "accept", otherwise it outputs "reject".

The following lemma shows that we have indeed obtained what we were after:

**Lemma 3.3** *Suppose the input $x \notin L$ and consider the probability, taken over the coin flips of $V_0^{IH}$, that there exists correct answers to both $v = 0$ and $v = 1$ in step 2 above. Let $\delta(k)$ be this probability, maximized over all possible strategies for the prover. Then $\delta(k)$ is negligible in $k$.*

**Proof** Consider a prover $P^*$ that plays an optimal strategy in order to have $V_0^{IP}$ accept $x \notin L$. This in particular means that we may assume that $P^*$ always sends a fixed first message $a$ which maximizes the number of $c$-values for which a correct answer exists. Thus the probability that this $P^*$ succeeds is $\delta(k)$. It now follows from Lemma 3.2 that if $\delta(k)$ is not negligible in $k$, there are infinitely many $k$'s where the fraction of $c$-values for which correct answers exist is larger than a polynomial fraction. This would contradict the soundness of $(P_0, V_0)$. $\square$

The part of $(P_0^{IH}, V_0^{IH})$ up to and including step 4 of the interactive hashing is called the *preamble* of $(P_0^{IH}, V_0^{IH})$.

As a tool in the main results, we will need the following protocol which describes a machine $M_0^{IH}$ that a prover can use to simulate the preamble of $(P_0^{IH}, V_0^{IH})$ on input any word $x$ and a private bit $b$ chosen in advance by the prover. This is done such that the prover can complete the protocol successfully in case the verifier's challenge $v$ equals $b$.

**The protocol** $(M_0^{IH}, V_0^{IH})$

1. $M_0^{IH}$ runs the honest verifier simulator $M_0$ for $(P_0, V_0)$ on input $x$ to obtain a conversation $(a, c, r)$. If this conversation is not an accepting conversation, $M_0^{IH}$ outputs ? and stops.

2. $M_0^{IH}$ executes the interactive hashing with $V_0^{IH}$ using the $c$ from the conversation just produced as the original choice. It names the two values $c_0, c_1$ isolated by the hashing such that $c = c_b$. Finally, $M_0^{IH}$ outputs $r$ and stops.

We have the following:

**Lemma 3.4** *Assume that, on input $x \in L$ and any bit $b$, we execute $(M_0^{IH}, V^*)$ where $V^*$ is any machine playing the role of $V_0^{IH}$. This always results in $M_0^{IH}$ outputting a correct answer to $v = b$. If $(P_0, V_0)$ is perfect, resp. statistical honest verifier zero-knowledge, then the amount of Shannon information that $V^*$ has about which $v$-value $M_0^{IH}$ can answer is 0 resp. negligible in $k$.*

**Proof** The Lemma follows from Lemma 3.1, the fact that the $c$-value used is uniform or statistically close to uniform, and the fact that when $x \in L$, first there exists a correct answer to any $c$-value and secondly $M_0$ always produces an accepting conversation. □

# 4 The Closure Property for $L$

We now prove Theorem 1.1. We are given a language $L$ in $HV3AM/PZKIP$ or in $HV3AM/SZKIP$, a family $F = \{f_n\}$ of monotone functions, an efficient secret sharing scheme $S$ for $F^*$, and an admissible parsing $\pi$. Let $x$ be a word parsed as $x_1, ..., x_n$, and let $A$ denote the set of indices $i$ such that $x_i \in L$. Its complement is denoted $A'$. Recall that $s_i$ denotes the $i$'th share produced by $S$, which is of length $l_i(n)$. We let $s_{i,j}$ denote the $j$'th bit of the $i$'th share. Our purpose is to exhibit a ZKIP that $f_n(A) = 1$.

### Overview

We first give an informal description of the proof system $(P, V)$ claimed by the theorem. Let's assume, for clarity, that all $l_i(n)$ are equal to 1.

The basic idea is that, for $x_i \in L$, the prover and the verifier engage in an execution of the preamble of $(P_0^{IH}, V_0^{IH})$, while for $x_i \notin L$, they execute $(M_0^{IH}, V_0^{IH})$. This is done in parallel for all $x_i$. In all $n$ cases, two challenge values are isolated. In case $x_i \in L$, the prover can answer both of them, while if $x_i \notin L$, he can answer at most one. This means that for all $i \in A'$, the prover is effectively committed to a bit, which challenge (i.e. which $v$-value) he can answer.

The verifier now selects a random bit $b_s$. The prover must then (on his own) complete all the conversations of $(P_0^{IH}, V_0^{IH})$ that were started earlier and send the results to $V$. All these conversations must be accepting. But in addition, we interpret the $v$-value answered in the $i$'th instance as the bit $s_i$ in an output of $S$. This determines a set of shares $s_1, .., s_n$, and $V$ will accept, only if this set of shares is consistent with the $b_s$ he chose.

Intuitively, this is sound since if too many $x_i$'s are not in $L$, i.e. $f_n(A) = 0$, we have $f_n^*(A') = 1$. This means $P$ will be committed to so many $s_i$'s, that a secret is already uniquely determined before $V$ chooses $b_s$, so $P$ can only survive with probability 1/2. It is zero-knowledge because it is easy to simulate if the 1-bit challenge from $V$ is known in advance, thus standard techniques apply.

### Proof of Theorem 1.1

Here follows a formal description of the protocol: Initially, both $P$ and $V$ parse the input $x$ using $\pi$ to obtain $n$ and $x_1, ..., x_n$. Then the following steps are repeated $k$ times:

1. First $P$ generates a set of shares $\{s_i | \; i \in A'\}$ distributed according to $D_{A'}$. This can be done by choosing a random bit $b_r$, running $S$ on input $n, b_r$ and discarding all shares that do not correspond to indices in $A'$. Now for $i = 1$ to $n$ do the following:

   If $i \in A$, $P$ executes $l_i(n)$ copies of the preamble of $(P_0^{IH}, V_0^{IH})$ with $V$ on input $x_i$.

   If $i \in A'$, then for $j = 1$ to $l_i(n)$, $P$ and $V$ run $(M_0^{IH}, V_0^{IH})$ on input input $x_i$. Here $s_{i,j}$ is $P$'s private input. If any instance of $M_0^{IHI}$ produces output ?, then $P$ sends ? and $s_i$ to $V$. $P$ stores the $l_i(n)$ answers $r_{i,1}, ..., r_{i,l_i(n)}$ produced privately.

2. $V$ chooses a random bit $b_s$ and sends it to $P$.

3. $P$ completes the set of shares $\{s_i | \ i \in A'\}$ to a full set of shares consistent with $b_s$. For each $i \in A$ he completes the $l_i(n)$ preambles of $(P_0^{IH}, V_0^{IH})$ by using for the $j$'th copy the bit $s_{i,j}$ as the $v$-value, and using the algorithm of $P_0^{IH}$ to compute a correct answer $r_{i,j}$.

   $P$ now sends to $V$ the complete set of answers $\{r_{i,j}\}$ and the set of shares $s_i$ that were not already sent in step 1.

4. For each $i$ where $P$ did not send ? in step 1, $V$ has now received $l_i(n)$ complete conversations of $(P_0^{IH}, V_0^{IH})$ with $x_i$ as input. $V$ checks that all these conversations would lead to accept by $V_0^{IH}$.

   $V$ has also received a complete set of shares $\{s_i | \ i = 1, .., n\}$. He checks that this set is consistent with $b_s$.

   If all checks are OK, $V$ accepts this iteration, otherwise he rejects and halts.

$V$ accepts, if and only if all $k$ iterations are accepted.

We now argue that this protocol has the required properties:

**Completeness**

- is trivial by completeness of $(P_0, V_0)$, by ,the properties of the simulator $M_0$, and by our assumptions on the sharing scheme $S$.

**Soundness**

There is nothing to prove, unless when parsed, the input word is of the form $x_1, ..., x_{n(k)}$ and $f_n(A) = 0$, where the index set $A$ is defined as in the protocol description. Consider an arbitrary prover $P^*$.

Consider the situation after step 1 of the protocol. For each index $i \in A'$, either the prover has sent $?, s_i$ or for each $j = 1, ..., l_i(n)$ he can answer one of the two challenges resulting from each execution of $(M_0^{IH}, V_0^{IH})$. By Lemma 3.3, in the latter case, the challenge that can be answered (and hence $s_{i,j}$) is uniquely determined, except with probability $\delta(k_i)$, where $k_i$ is the length of $x_i$. Therefore, after step 1, the set of shares $\{s_i | \ i \in A'\}$ that the verifier will accept in step 4 is uniquely determined from the preambles or because the prover sent $?, s_i$ immediately, except with probability at most $\epsilon(k) := \sum_i l_i(n)\delta(k_i)$. Since the parsing is admissible, all $k_i$'s are polynomial in $k$, and hence $\epsilon(k)$ is negligible in $k$.

Now note that $f_n(A) = 0$ implies $f_n^*(A') = 1$, by the definition of dual functions, and that therefore any set of shares $\{s_i | \ i \in A'\}$ can be consistent with at most one value of the secret. Hence the probability that $V$ accepts

$k$ iteration of steps 1-4 is at most $(1/2 + \epsilon(k))^k$, which is clearly negligible in $k$.

**Zero-Knowledge**

We describe a simulator that works against any verifier $V^*$.

Repeat the following $k$ times:

1. Start by choosing a random bit $b$ and run $S$ on input $n, b$ to get shares $s_1, ..., s_n$.

2. For $i = 1$ to $n$ do the following:

   For $j = 1$ to $l_i(n)$, start $(M_0^{IH}, V_0^{IH})$ on input $x_i, s_{i,j}$. If any of the copies produce ? as output, send ? and $s_i$ to $V^*$. Otherwise, let each copy execute the preamble of $(P_0^{IH}, V_0^{IH})$ with $V^*$. Save all answers $r_{i,j}$ produced.

3. Receive $b_s$ from $V^*$.

4. If $b = b_s$, send to $V^*$ all answers $r_{i,j}$, all shares $s_i$ not yet sent, and stop simulation of this iteration.

   Otherwise, rewind $V^*$ to its state at the start of step 1 and go to step 1.

To show that this simulation works, consider first the case where the honest verifier simulation of $(P_0, V_0)$ is perfect. Note that we are required to simulate correctly only in cases where the input is in $F(\pi, L)$. This means that $f_n(A) = 1$ and hence $f_n^*(A') = 0$, using the notation from above. By Lemma 3.4, $V^*$ receives no information about $s_i$ where $i \in A$, and hence since $f_n^*(A') = 0$ receives also no information on $b$. Hence $V^*$'s choice of $b_s$ is uncorrelated to $b$ so that the probability that $b = b_s$ is $1/2$, and the complete simulation takes expected linear time.

For correctness of the output distribution, note that for $i \in A'$, the simulator follows completely the prover's algorithm. For $i \in A$, consider the simulation of a single preamble. The $a$ sent initially has exactly the right distribution. In the interactive hashing, both simulator and prover answer consistently with an original choice $c$ with uniform distribution, so all message sent there also have the right distribution. The event that $b = b_s$ is uncorrelated to all messages considered so far (by Lemma 3.4), so the fact that we wait until $b = b_s$ does not bias the distribution actually output. Finally, the answers $r_{i,j}$ produced by $P$ resp. the simulator are in both cases random samples of what the original $P_0$ would produce, given the initial conversation defined in the preamble.

This finishes the perfect zero-knowledge case. In the statistical zero-knowledge case, we use the same simulator, except that the oracle we have for making conversations of $(P_0, V_0)$ (the honest verifier simulator) now generates output that is statistically close to the perfect case. Hence the output we produce will also be statistically close to perfect.

## 5 The Closure Property for $L'$

We now prove Theorem 1.3 and Corollary 1.4. The notation is the same as in the previous section. Note, however, that we now have a secret sharing $S$ for $F$ (and not $F^*$).

### Overview

First the verifier $V$ will (privately) choose a bit $b_s$ and will share it using the secret sharing scheme to get shares $s_1, .., s_n$. Then for each $i = 1..n, j = 1..l_i(n)$ the verifier $V$ and the prover $P$ will run $(M_0^{IH}, V_0^{IH})$ on input $x_i$, BUT this time the verifier will play the role of $M_0^{IH}$. $V$ uses the bit $s_{i,j}$ as private input for the $(i,j)$'th instance. The verifier $V$ is now able to answer correctly $v = s_{i,j}$.

Then the prover must guess the bit $b_s$, and $V$ rejects if the guess is incorrect.

Note that for $i$'s where $x_i \in L$ there is a correct answer in $(P_0, V_0)$ to any challenge value, so the execution of $(M_0^{IH}, V_0^{IH})$ tells $P$ nothing about $s_{i,j}$, whereas if $x_i \notin L$, $s_{i,j}$ is most likely uniquely determined from the conversation. Hence, intuitively, the protocol is complete since if enough $x_i$'s are in $L'$ (i.e. $f_n(A') = 1$), $P$ can find enough $s_i$'s to be able to compute $b_s$. It is sound since if too few $x_i$'s are in $L'$ (i.e. $f_n(A') = 0$), $P$ gets only negligible information about $b_s$. It is honest verifier ZK, since the honest verifier knows in advance what the provers answer will be.

### Proof of Theorem 1.3 and Corollary 1.4

Initially, both $P$ and $V$ parse the input $x$ using $\pi$ to obtain $n$ and $x_1, ..., x_n$. Then the following steps are repeated $k$ times:

1. $V$ chooses a random bit $b_s$ and runs $S$ on input $n, b_s$ to get shares $s_1, ..., s_n$.

2. For $i = 1$ to $n$ do the following:

   For $j = 1$ to $l_i(n)$, $V$ and $P$ run $(P_0^{IH}, V_0^{IH})$ on input $x_i$. $V$'s private input is $s_{i,j}$. If any of the copies produce ? as output, send ? and $s_i$ to $P$.

3. For $i = 1$ to $n$ do the following:

   If $x_i \notin L$, $P$ uses the messages received in the $i, j$'th preamble executed with $x_i$ as input to determine $s_{i,j}$. Then $P$ knows $\{s_i | \ i \in A'\}$. He uses these shares to determine a secret (bit) $b$ and sends it to $V$. If the shares are inconsistent or could not be determined, $P$ sends a random bit $b$.

4. $V$ checks that $b = b_s$ and accepts this iteration if this is case, otherwise he rejects and halts.

$V$ accepts, if and only if all $k$ iterations are accepted.

We now argue that this protocol has the required properties:

**Completeness**

Consider an $i \in A$, i.e. $x_i \notin L$. For such an $i$, either $s_i$ is sent to $V$ directly or is with overwhelming probability uniquely determined from the preambles executed on $x_i$,by Lemma 3.3. Therefore $P$ can compute the shares $\{s_i | \ i \in A'\}$ (although this may require unbounded computing power). We assume here that the input is in $F(\pi, L')$, which means that $f_n(A') = 1$. Therefore $b_s$ is uniquely determined from $\{s_i | \ i \in A'\}$ and hence $P$ can answer correctly, except with negligible probability.

**Soundness**

There is nothing to prove, unless the input word when parsed is of the form $x_1, ..., x_n$ and $f_n(A') = 0$. An arbitrary $P^*$ may be able to compute the shares $\{s_i | \ i \in A'\}$, but these contain no information on $b_s$ since $f_n(A') = 0$. By Lemma 3.4 the preambles executed on $x_i \in L$ give at most a negligible amount of information on the shares, and therefore $P^*$ cannot guess any $b_s$ with probability significantly different from 1/2. For $k$ iterations the acceptance probability therefore becomes negligible in $k$.

**Honest Verifier Zero-Knowledge**

The simulation is straightforward: simply follow the honest verifier's algorithm and use the value of $b_s$ (which you know from verifier's random tape) as the prover's answer $b$. This is statistically close to the real conversation as the simulation of all messages but the last one (the $b$) is perfect, and we have shown that the real prover's $b$-values are correct except with negligible probability.

To show Corollary 1.4, notice that the assumptions imply, by results in [5], that an unconditionally hiding bit commitment scheme exists. Using the method from [10] such a bit commitment scheme can be used to transform any honest verifier statistical zero-knowledge protocol into one that is statistical zero-knowledge in general.

## 6 Extentions

We have assumed for simplicity that only one basic language $L$ is involved in our construction. But our method generalizes trivially to proving in statistical ZK statements such as $x_1 \in L_1 \vee ... \vee x_n \in L_n$ where $L_i \in HV3AM/SZKIP$, put differently, our results still hold, even if each statement on a subword refers to a different language. It is also possible to mix to some extent languages and their complements. For example, the statement $x_1 \in L_1 \vee x_2 \notin L_2$ can be proved in statistical ZK using a straightforward generalization of a corresponding method from [7].

## 7 Acknowledgement

Thanks to Oded Goldreich for many constructive suggestions for an improved presentation of our results.

## References

[1] M.Bellare and M.Yung: *On certifying cryptographic tools: the case of trapdoor permutations*, Proc. of Crypto 92.

[2] J. Benaloh and J. Leichter: *Generalized Secret Sharing and Monotone Functions*, Proc. of Crypto 88, Springer Verlag LNCS series, 25–35.

[3] M.Blum, A. De Santis, S.Micali and G.Persiano: *Non-Interactive Zero-Knowledge*, SIAM J. of Computing, vol.20, 1991.

[4] R.Cramer, I.Damgård, and B.Schoenmakers: *Proofs of partial knowledge and simplified design of witness hiding protocols*, proc. of Crypto 94.

[5] I.Damgård: *Interactive Hashing can Simplify Zero-Knowledge Protocol Design Without Computational Assumptions*, Proc. of Crypto 93.

[6] L.Fortnow: *The complexity of perfect zero-knowledge*, Adv. in Comp. Research, vol.5, 1989, pp.327-344.

[7] A.De Santis, G Di Crescenzo, G. Persiano and M.Yung: *On Monotone Formula Closure of SKZ*, Proc. of FOCS 94.

[8] S.Goldwasser, S.Micali and C.Rackoff: *The Knowledge Complexity of Interactive Proof Systems*, SIAM J.Computing, Vol.18, pp.186-208, 1989.

[9] J.Van de Graaf and R.Peralta: *A simple and secure way to show the validity of your public key*, Proc. of Crypto 87.

[10] R.Ostrovsky, Venkatesan and M.Yung: *Interactive Hashing Simplifies Zero-Knowledge Protocol Design*, Proc. of EuroCrypt 93.

[11] M.Naor, R.Ostrovsky, Venkatesan and M.Yung: *Zero-Knowledge Arguments for NP can be Based on General Complexity Assumptions*, Proc. of Crypto 92.

[12] M. Naor and M. Yung, *Universal one-way hash functions and their cryptographic applications*, Proc. 21st Annual ACM Symp. Theory of Computing, 1989, pp. 33–43.

[13] R.Raz and A.Wigderson: *Monotone circuits for matching require linear depth*, J. ACM, vol.39, 1992, pp.736-744.