Non-Interactive Circuit Based Proofs and Non-Interactive Perfect Zero-knowledge with Preprocessing

Ivan Damgård

Aarhus University, Mathematical Institute
Ny Munkegade, DK 8000 Aarhus C, Denmark

Abstract. In the first part of this paper, we present a non-interactive zero-knowledge proof system for Circuit Satisfiability. With this protocol, we can prove an arbitrary NP-statement non-interactively without using Karp-reductions to 3-SAT or Graph Hamiltonicity. The proof system is based on the quadratic residuosity problem and allows processing of XOR and NOT gates at virtually no cost. It is significantly more efficient than previously known non-interactive proof systems. In the second part, we present protocols based on the existence of collision intractable hash functions, leading to a *statistical* zero-knowledge non-interactive argument with preprocessing for *any* NP-statement. Under the certified discrete log assumption, the protocol is perfect zero-knowledge. In the preprocessing, the parties need only exchange messages of length independent of the theorem to be proved later. This is the first protocol with such efficient preprocessing that does not need to assume oblivious transfer. Finally we present a perfect zero-knowledge non-interactive protocol based on discrete logarithms that may potentially remove the need for preprocessing.

1 Introduction

A non-interactive zero-knowledge proof system is a protocol that allows a prover to convince a verifier that some statement is true, simply by sending one message to the verifier. The prover should be unable to cheat, and the verifier should learn nothing more from receiving the message than the mere fact that the statement involved is true. Accomplishing this for non-trivial statements requires that prover and verifier share a random string, the randomness of which they both trust.

Such proof systems were introduced in [BFM], [BSMP], where a proof system for 3-SAT was presented, based on the Quadratic Residuosity Assumption (QRA). Later, a proof system for Graph Hamiltonicity (GH) was presented in [LS], which was based on the more general assumption that a one-way permutation exists, or if the prover has only polynomial computing power, that one-way trapdoor permutations exist.

There has also been some work published on non-interactive zero-knowledge with preprocessing [SMP],[KMO], in which prover and verifier first execute an interactive phase, which allows the prover to later convince the verifier about some statement. This statement may be unknown when the preprocessing is done. The proofs in [SMP] were based on any one-way function, while [KMO] needed in addition oblivious transfer, but had a much smaller preprocessing step.

Since both 3-SAT and GH are NP-complete (and since sufficiently nice reductions to them are known), the results of [BSMP] and [LS] imply that any NP-statement can be proved in non-interactive zero-knowledge, if one-way trapdoor permutations exist, or in particular if quadratic residuosity decision is hard.

This, however, requires the use of Karp-reductions, such that from the original problem instance, one constructs for example a graph that is Hamiltonian, precisely if the original instance was a yes-instance. Thus the size of theorems one has to work with becomes larger than the original instance, implying a loss of efficiency.

A proof system for SAT, such as the one presented here, does not have this problem: for any NP-language L, by Cooks Theorem, there is a circuit C_L that when given a word w as fixed input on some of the wires, is satisfiable precisely if $w \in L$. Thus this circuit can be used directly in the proof system. Moreover, we are free to build ad hoc as small a circuit as possible for the given problem. Such a circuit will usually be far smaller than one conctructed from Cook's theorem. Although this circuit could also be handled by constructing from it a (linearly larger) 3-SAT instance and using the protocol of [BSMP], using our protocol directly will be significantly more efficient: we need the same amount of work and shared random bits per binary gate as the 3-SAT protocol needs per clause, and moreover all XOR and NOT gates can be handled in our protocol at virtually no cost (1 multiplication and no bits of the shared random string).

The known non-interactive zero-knowledge proofs for NP-complete problems, with or without preprocessing, are only computationally zero-knowledge. This is because the results were proved in the model where the prover may have unlimited computing power, while the verifier is polynomially bounded. In this model, perfect/statistical ZK seems to require special properties of the statement proved: if we were given a non-interactive perfect ZK proof for an NP-complete problem, we could use an unconditionally hiding bit commitment scheme (see USBCA, section 4) to "move" the proof system to the ordinary interactive model. Thus, the "impossibility" result of [Fo] would apply.

We therefore consider in stead the dual model (following [BCC]), where the verifier may be unlimited, while the prover must be polynomially bounded, i.e. we consider ZK arguments. We present a non-interactive statistical ZK argument with preprocessing for GH. This scheme will work if collision intractable hash functions exist, and has the property that the interaction required in the preprocessing phase does not depend on the size of theorem to be proved later. An alternative protocol for this problem could be derived from the techniques in [KMO] or [SP], but this would require an oblivious transfer subprotocol ([SP] gives a specific example based on the quadratic residuosity assumption), and oblivious transfer is not known to be (and probably is not) implementable based only on collision intractable hash functions.

Finally, we look at the problem of constructing a perfect (or statistical) zero-knowledge argument in the shared string model for an NP-complete problem. [SP] considered this problem and solved it in a modified scenario where, in addition to the shared random string, the prover has access to a message broadcast by the verifier, specifying a public key to use in the proof. To the best of the authors knowledge, the problem in the original shared string model is still open. We present a potential solution based on discrete logs that is provably perfect zero-knowledge, and is conjectured to be a proof system.

2 Notation and Definitions

The protocols we describe take place between a prover P and a verifier V. These are probabilistic Turing machines. In Section 3, the prover may have infinite computing power (but is not required to use it in the protocol), while the verifier is polynomially bounded. The roles are reversed in Section 4.

In Section 3, the *shared random string model* is assumed, i.e. both prover and verifier have read-only access to a one-way infinite tape containing independent random bits. This bit string is called α. P tries to convince V that some word w is in the NP-language L.

By $A(\cdot)$, we denote the random variable resulting from running probabilistic algorithm A on input $\cdot$. Thus $\sigma = P(w, \alpha)$ is the proof computed by P and sent to V. $V(\sigma, \alpha)$ is either *accept* or *reject*, and the proof system is said to accept or reject accordingly.

(P, V) is said to be a *proof system* for L, if it is *complete*: $w \in L$ implies that $Prob(V(w, \sigma, \alpha)) =$ *reject* is superpolynomially small in $|w|$; and *sound*: whenever $w \notin L$, for any (possibly infinitely powerful) P^*, $Prob(V(P^*(\alpha, w), \alpha)) =$ *accept* is superpolynomially small in $|w|$.

(P, V) is said to be *zero-knowledge* if there is a probabilistic polynomial time simulator M, which on input w only produces a random string and "proof" that looks just like the strings actually used in the protocol. More precisely, $M(w)$ is polynomially indistinguishable from α, σ (the simulator does not have to be relative to a verifier, because V does not take part in any interaction).

What we have described here is in fact what is called *bounded* non-interactive zero-knowledge in [BSMP].

In Section 4, we change the model, such that the verifier may now have infinite computing power, while the prover is polynomially bounded. Moreover, P and V are now interactive Turing machines, and there is no shared random string. By $(P, V)(w)$, we denote the verifiers decision after talking to P about common input w. In addition to w, P gets access to a witness x proving that $w \in L$.

The definition that (P, V) is a proof system for L is the similar to the above: the proof system must be *complete*: if $w \in L$, and x is a valid witness for w, then $Prob((P, V)(w) =$ *accept*$)$ is superpolynomially large in $|w|$. It must be *sound*:

if $w \notin L$, then for any probabilistic polynomial time P^*, any auxiluary input x, and any set of coinflips for P^*, $Prob((P^*, V)(w) = accept)$ is superpolynomially small in $|w|$.

The definition of zero-knowledge in this model is the same as the standard one for interactive proof systems, except that we allow infinitely powerful verifiers, and hence are only interested in statistical or perfect zero-knowledge.

In addition, we would like the system to satisfy an additional requirement - (P, V) is said to be a *proof system with preprocessing*, if the protocol can be split in an interactive preprocessing phase, where neither party is given access to w, and a subsequent non-interactive proof-phase, where P gets w and x, while V gets only w. From this P computes in polynomial time a proof, which is sent to V. V checks the proof against w and the conversation from the preprocessing phase, and outputs *accept* or *reject*. Although the protocol should be secure against an infinitely powerful V, we will only look at protocols that a "real-life" verifier can execute, so we require that V can do both preprocessing and checking in polynomial time.

Note that w, x may be chosen as a function of the interaction that took place in the preprocessing phase.

3 Non-Interactive Proof System for SAT

In this section, we describe non-interactive proofs that are bounded in the sense of [BSMP]: the prover can only prove 1 SAT-instance of length $O(\sqrt[3]{l})$, where l is the length of the shared random string. In [BSMP], methods are given that allow transformation of their protocol to one where the prover can prove arbitrarily many instances of any size. These methods are also applicable to our protocol, with only trivial modifications.

The common input to prover and verifier is a one-output satisfiable Boolean circuit C. The prover may be thought of as a polynomial time machine, that knows an assignment of bits to the input wires that causes the output to be 1.

Without loss of generality, assume that C has gates $G_1, G_2, ..., G_k$, where all gates have two inputs and 1 output (which may be fanned out to several other gates). Thus each gate G_i may be described by T_i, a binary array with 4 rows and 3 columns, the truthtable of G_i. The gates receiving 1 or 2 input bits to C are called input gates. The circuit may specify that an input bit should enter several input gates. Such input bits are said to be shared between the gates.

The proof will use the quadratic residuosity assumption (QRA, introduced in [GM]) which says that for an n which is the product of two large primes, when given a random x with Jacobi symbol 1, it is hard to tell whether x is a quadratic residue, if the factorization of n is not known. We will say that x *hides the bit* b, if the quadratic character of x equals b. Clearly x_1 and x_2 hide the same bit precisely if x_1x_2 is a square modulo n. If y is a non-square, then x hides b exactly if y^bx is a square. When P includes a square root of y^bx in his proof, this is referred to as *opening the number* x.

We can now describe the algorithm of P, which works by combining techniques from [BCC] and [BSMP]:

PROVER'S ALGORITHM

1. Choose random primes p, q, such that $n = pq$ is of length k bits, and a random non-square y.
2. From the first k^3 bits of α, produce a proof that y is a nonsquare modulo n, that n has two distinct prime factors, and that n is not a perfect square, exactly as in [BSMP].
3. Split the first $264k^3$ bits of the remainder of α in segments of length k bits, and consider the segments as $264k^2$ integers. Discard all numbers that are $\geq n$, or are $< n$, but have Jacobi symbol -1. Compute the bits hidden by all the remaining numbers.
4. The obtained sequence of numbers is split in groups of 3. A group x_1, x_2, x_3 is said to encode a row in the truth table T if T has a row b_1, b_2, b_3 such that x_i hides b_i for $i = 1, 2, 3$.
5. For $i = 1...k$ do the following:
   repeat for the first $11k$ unused groups in the sequence:
   If the group does not encode a row in T_i, prove this by opening all numbers in the group. Otherwise write information in the proof that this group encodes some row in T_i.
   Finally divide all the unopened groups assigned to T_i into 4 classes, such that all groups in a class encode the same row in T_i (only the subdivision is included in the proof, P does not reveal which row corresponds to which group). For each pair of groups belonging to the same class, prove this by displaying square roots of products of corresponding numbers in the groups.
6. The computation in C induced by using the satisfying assignment as input will select a row in each T_i. Now include in the proof a pointer to a group encoding this row. Let x_{1i}, x_{2i}, x_{3i} be this group, hiding the first and second input bit, resp. the output bit.
   For $i = 1..k$, do:
   If for some j, T_j receives its t'th input from T_i, prove consistency of the computation by displaying a square root of $x_{3i}x_{tj}$.
   If T_i is an input gate such that its t'th input bit is shared with the m'th input bit of gate T_j, where $j > i$, prove consistency by displaying a square root of $x_{ti}x_{mj}$.
   If T_i is the final output gate of C, prove that C outputs 1 by opening x_{3i}.

Note that this algorithm fails, if there are not $11k^2$ groups available for step 5, or if encodings of all 4 rows do not show up in step 5. In these cases P outputs a random string and stops. We proceed by describing how V should check the proof:

VERIFIER'S ALGORITHM

i. As in [BSMP], verify the proof that y is a non-square modulo n, that n has two distinct prime factors, and that n is not a square.
ii. Check that all numbers discarded by P in step 3 are $\geq n$ or have Jacobi symbol -1.
iii. For $i = 1..k$, check that P has correctly opened all numbers in groups discarded in step 5, and that indeed no opened group encodes a row of T_i.
iv. Check all the other sub-proofs produced by P in step 5. Check that the unopened groups assigned to each T_i have been divided into 4 classes.
v. Check all the square roots produced by P in step 6.

The first result about this protocol is:

Theorem 1 (P, V) is a non-interactive proof system for SAT.

Proof Completeness is obvious by inspection of the protocol: P only fails in the two cases mentioned after step 6, and their probability is exponentially small in k: for a fixed n, each k bit integer has a chance of at least 1/4 of not being discarded, so we expect to have $66k^2$ integers left after step 3. It follows from Bernsteins law of large numbers that the probability that the actual number of remaining integers is less than half the expected value is exponentially small in k^2. Since there are at most 2^k possible n's, the probability that there exists an n leading to less than $33k^2$ remaining integers is exponentially small in k. The other failure case is handled below

For soundness, we have to argue that given C is not satisfiable, P^* manages to convince V with at most negligible probability. Assume that P^* has in fact produced a convincing proof.

First, by the proof in [BSMP], we may assume that the n, y given by P^* has the correct form. We may also assume that there were $11k^2$ groups available for step 5 of the prover.

Now observe that if C is not satisfiable, at least for one i, the group G used for T_i in step 6 does not encode a row of T_i. But this group comes from one of the classes of step 5. G cannot be proven equivalent to any of the proper encodings, so since there are only 4 classes and no proper encodings were opened, at least one row encoding of T_i never showed up during step 5.

Since α is random, for each given choice of n, y, the bits hidden by the numbers of Jacobi symbol 1 are independent and each bit is 1 with probability 1/2. Therefore each group considered encodes each of the possible 8 rows with probability 1/8. By elementary probability theory, we find that the probability that encodings of all possible 8 rows did not show up for one of the k gates is at most $8k(7/8)^{11k}$. Since there are at most 2^{2k} possibilities for n, y, the probability that there exists a choice of n, y allowing cheating by P^* (or failure for the honest prover) is at most

$$2^{2k} 8k(\frac{7}{8})^{11k} \leq 8k(0.93)^k$$

□

Theorem 2 Under QRA, (P, V) is a zero-knowledge non-interactive proof system.

Proof The simulator may be constructed as follows: first choose a modulus n of the correct form and a random *square* y, and then produce a simulated "proof" that y is a non-square modulo n, exactly as in [BSMP]. This will also produce a first segment of the simulated shared random string.

The rest of the shared random string is produced as follows: the simulator chooses a sufficient number of random bits, splits them in k bit integers, and marks as discarded those that are $\geq n$ or have Jacobi symbol -1. The rest of the integers are all replaced by random squares modulo n. This concludes the computation of the simulated shared random string. The simulator includes pointers to the discarded numbers in its proof, just as the prover would have done.

Note that since y is chosen as a square, the simulator can "open" each of the squares now produced, both as a 1 and as a 0.

The simulator now makes groups of the non-discarded numbers, and assigns $11k$ groups to each truth table, as in the prover's step 5. For each group, it decides with probability $1/2$ to "open" the group as a random group that does not properly encode a row in the truthtable in question. With probability $1/2$ it decides to leave the group unopened. Each unopened group is randomly put into 1 of 4 classes. If each truthtable does not own at least 1 group in all 4 classes, we fail and stop, as the prover would. Otherwise, for each truthtable, we "prove" equivalence of groups in the same class by displaying square roots of products of corresponding numbers in the groups. This is easy, since they are all squares.

Finally we simulate the prover's step 6 by choosing for each truthtable a random unopened group. We then display square roots of products of some of the numbers in the rows, exactly as required in step 6. Finally, we "open" the output bit as a 1.

It is easy to see that the only difference between the simulation and the real proof lies in the distribution of bits that are hidden in unopened numbers. Hence it is intuitively reasonable that a successful distinguisher would need the ability to distinguish squares from nonsquares.

This can be proved by contradiction: assume that for infinitely many k, there exists a satisfiable circuit C of size k for which the proofs constructed by our protocol are efficiently distinguishable from the simulation. We can then derive a contradiction with QRA by constructing an efficient algorithm A which on input n, y and a satisfying assignment for C will produce an output satisfying the following:

- If y is a square modulo n, the output is distributed as the simulator's output.
- If y is a non-square, the output is distributed exactly as the real random shared string and the prover's proof.

It is clear that this, together with the distinguisher, leads to a nonuniform algorithm violating QRA (non-uniform because we have to hardwire in the satisfying assignment for C).

To construct A, we proceed as follows: first run the simulation from [BSMP] of the proof that y is a nonsquare modulo n. It is proved in [BSMP] that this part has the property we require of A. We then construct the last part of the shared random string as in the simulator's algorithm above, EXCEPT that after constructing the non-discarded numbers as random squares, we decide at random for each number whether or not to multiply it by y. This means that A now can open each resulting number in only 1 way. But since A knows a satisfying assignment for C, A can still complete the proof: it simply follows the prover's algorithm. It should now be clear that if y is a square, nothing is changed compared to the simulation (all non-discarded numbers are still squares), while otherwise we get exactly the real prover's situation (the non-discarded numbers hide independent random bits). Note that, as mentioned in [BSMP], the verifier will obtain an indistinguishable view, no matter which satisfying assignment the prover uses. It therefore does not matter whether A uses the same assignment as the prover$\square$

The protocol we have described is capable of handling arbitrary binary gates. However, from the homomorphic property of the mapping from numbers modulo n to the bits they hide, it is clear that XOR and NOT gates can be handled much more efficiently than general binary gates: given two numbers hiding input bits to an XOR gate, we simply multiply the numbers modulo n to get a bit hiding the output. Similarly, given a number hiding an input bit to a NOT gate, we multiply by y.

4 Perfect and Statistical Zero-Knowledge Arguments With Preprocessing

In this section we will be concerned with constructing a non-interactive perfect or statistical zero-knowledge protocol for an NP-complete problem (both with and without preprocessing). To get perfect zero-knowledge, however, it seems we have to change to the model where the prover is polynomially bounded, whereas the verifier may be unlimited, so we get zero-knowledge arguments. For ordinary interactive proofs, this is because of Forthnow's "impossibility result". The same result can be applied to non-interactive proofs, provided that unconditionally hiding bit commitments exist (assumption USBCA below).

One example of such commitments is the one presented in [CDG] and independently in [BKK]. Here, the prover receives a prime p, the factorization of $p-1$, a generator g of Z_p^*, and a random element $a \in Z_p^*$. The prover can check from these data that indeed g generates Z_p^*. A commitment to the bit b is computed as $a^b g^r$ where r is chosen uniformly in $[0..p-2]$. The prover is unable to change his mind unless he can compute the discrete log base g of a. On the other hand, commitments have distribution independent of the bits they hide. We will refer to this scheme as the discrete log scheme (DLS). The assumption that P cannot find the discrete log of a, even when given the factors of $p-1$ is known as the certified discrete log assumption (CDLA).

Rather than using a particular commitment scheme, it is natural to try to base the results on the general assumption that there exists a bit commitment scheme hiding bits unconditionally, since there is evidence to suggest that this is the minimal assumption that will support perfect zero-knowledge arguments in general [Da]. More precisely, we consider the following assumption:

Unconditionally Secure Bit Commitment Assumption (USBCA) There exists a infinite family of finite sets $\{I_k\}$, where an element in I_k is a function $BC : \{0,1\} \times \{0,1\}^k \rightarrow \{0,1\}^{s(k)}$, and $s(k)$ is polynomially related to k. The following should be satisfied:

- Given k, a random element (instance) of I_k can be selected in probabilistic polynomial time.
- Given $BC \in I_k$ selected according to the above condition, no probalistic polynomial time algorithm can find r, r' such that $BC(1,r) = BC(0,r')$.
- For any instance BC, the distribution of $BC(1,r)$ equals the distribution of $BC(0,r)$, when r is unformly chosen.

In general, establishing and opening a commitment may be possible by some interaction between sender and receiver. Such schemes are not usable in this context, however.

In the third condition above, we may replace the requirement that the two distributions are equal with the requirement that they be statistically indistinguishable.

USBCA follows from many different intractability assumptions: hardness of discrete log, factoring or graph isomorphism; the existence of perfect zero-knowledge MA-proofs of knowledge [Da]; or hardness of some forms of the knapsack problem [NI].

To get more efficient protocols, we will need another assumption, namely that families of collision-intractable (or collision free) hash functions exist. Such a family has the property that it is easy to select at random a member function h with a k bit output, but although the input is longer than k bits, it is hard to find $x \neq y$ such that $h(x) = h(y)$. See [Da2] or [NY1] for a formal definition. The assumption on existence of collision intractable hash functions is at least as strong as USBCA, by a result of Naor and Yung [NY1]:

Proposition The existence of families of collision-intractable hash functions imply USBCA.

If the function $s(k)$ from USBCA is at most k, it is easy to see using the tecniques from [Da] that the inverse implication holds. Hence, for example, CDLA implies existence of collision intractable hash functions. The general inverse implication seems likely to be true also, but this is an open problem so far.

In the next two subsections, we look at protocols with preprocessing, and return to the shared string model in Section 4.3.

4.1 The protocol by Shamir and Lapidot

A simple approach to constructing an interactive argument with preprocessing for an NP-complete problem is to use the technique of Lapidot and Shamir [LS], which will give us a protocol for GH: they present a proof system with preprocessing which can be adapted to be based on USBCA.

For convenience, we briefly repeat the protocol here: in the preprocesing phase, P commits to k $k \times k$ incidence matrices $H_i, i = 1...k$, that each represent a graph consisting of one random Hamiltonian cycle. V responds with k random bits $b_1, ..., b_k$. In the proof phase, P sends a k-node graph G that he wants to prove is Hamiltonian, and also opens completely all those H_i for which $b_i = 0$, so that V can check that they were correctly constructed. Finally, for those i, where $b_i = 1$, P shows a permutation of the rows and columns of H_i and opens as a 0 all those entries in the permuted matrix that corresponds to 0's in the incidence matrix of G.

4.2 An Improvement Based on Hash Functions

One disadvantage with the protocol of Lapidot and Shamir is that it only works if G happens to have k nodes exactly, which is a problem if P does not know at preprocessing time which graph he will want to prove later. A simple solution is to let P supply sets of H's with j nodes for $j = 1..K$, where K is some maximum polynomially related to the security parameter k. Although this is still polynomial in k, it is hardly an attractive solution, since P and V have to exchange a number of bits corresponding to the value of K, even if G turns out to be much smaller.

Based on the assumption that collision intractable hash functions exist, we propose a different way to use the techniques of [LS], for which a much more efficient preprocessing and proof phase is possible. More precisely:

- The communication complexity of the preprocessing phase is independent of both K and the size of G.
- The communication complexity of the proof phase is $O(j^2ks(k) + jk)$ bits, where j is the size of G and k is the security parameter. In particular, it is independent of K, and if G has k nodes, we get essentially the same complexity as the basic Lapidot/Shamir solution.

First, let us remark that USBCA implies that one-way functions exist: consider the function that maps r, b to $BC(r, b)$. This function must be hard to invert, since the prover could clearly use an inversion algorithm to cheat. Thus, by [Na], USBCA also implies the existence of a bit commitment scheme with the dual property: the committer unconditionally cannot cheat, but the receiver of a commitment may find the bits if he has enough computing power.

In [FS], a construction is presented which allows transforming any bit commitment scheme into one that is *chameleon*, assuming that a one-way function exists; i.e. using this result, we may assume that associated with an instance of the commitment scheme is some *trapdoor information*, which can be chosen by the verifier, and which allows changing the contents of commitments.

The final observation we need comes from [Da2], where it is shown how to construct a collision intractable hash function h that is defined for arbitrary length inputs, based on a collision intractable function f from m bits to k bits, where $m > k$. We repeat here a simplified version of the construction which will be sufficient for our purposes. Put $t = m - k$. To hash input M, split it in t-bit blocks $M_1, ..., M_n$, padding the last block with 0's if needed. Then we define:

$$h(M, Z) = f(M_n || f(M_{n-1} || f(\cdots f(M_2 || f(M_1 || Z)) \cdots))),$$

where Z denotes a string of k bits, and $||$ denotes concatenation. It is now quite easy to prove that if it is infeasible to find collisions for f, then it is infeasible to find M, M', Z, Z' such that $M \neq M'$, the length of M equals that of M', and $h(M, Z) = h(M', Z')$. Moreover, it always holds that

$$h(M, Z) = h(M_j || M_{j+1} \cdots || M_n, h(M_1 || \cdots || M_{j-1}, Z))$$

In other words, when the prover sends a hashvalue, this commits him to the preimage, but the construction of h allows him to convincingly reveal only part of it at some later time. For simplicity in the following, we will assume that $t = k$. It is easy to modify the construction to do without this condition.

Let $H^{(j)} = \{H_i | i = 1...k\}$ denote a set of matrices chosen by the prover as in the above description, where k is the security parameter of the protocol below, and let $BC(H^{(j)}, R)$ denote a set of commitments to the bits in $H^{(j)}$, computed with random input R. The preprocessing goes as follows:

PREPROCESSING PHASE

1. V chooses an instance of a chameleon bit commitment scheme BC according to USBCA and the above observations. Using commitments based on a one way function, he convinces P in zero-knowledge that he, V, knows the trapdoor for the commitment scheme (using for example the general protocol from [BCC]).
   He also chooses a random member h of a family of collision intractable hash functions (constructed as above), and sends h to P.
2. P chooses at random $H^{(j)}$ for $j = 1...K$. He then computes
$$A_j = h(BC(H^{(j)}, R_j), Z_j)$$
   for $j = 1...K$ and randomly chosen R_j, Z_j. Finally, he sends to V the hashed image of the A_j's $h(A_K || \cdots || A_1, Z)$, for randomly chosen Z.
3. V returns a string E consisting of k random bits.

PROVER'S ALGORITHM, PROOF PHASE

1. Given a Hamiltonian graph G with j nodes, where $1 \leq j \leq K$, the prover sends $A_1, ..., A_j$, $BC(H^{(j)}, R_j)$, Z_j, and $h(BC(H^{(j+1)}, R_{j+1})|| \cdots ||BC(H^{(K)}, R_K), Z)$.
2. From $BC(H^{(j)}, R_j)$ and E, the prover generates a proof that G is Hamiltonian, as described above.

VERIFIER'S ALGORITHM, PROOF PHASE

1. Given a Hamiltonian graph G with j nodes, where $1 \leq j \leq K$, and the data received from the prover, the verifier checks that $A_j = h(BC(H^{(j)}, R_j), Z_j)$, and that $h(A_j||A_{j-1} \cdots ||A_1, h(A_K|| \cdots ||A_{j+1}, Z))$ equals the hash value received in the preprocessing.
2. Use $BC(H^{(j)}, R_j)$ and E to verify the provers proof as described above.

We can now prove:

Theorem 3 Under the assumption that families of collision intractable hash functions exist, the above constitutes a statistical zero-knowledge non-interactive argument with preprocessing for GH.

Proof sketch Completeness is (as usual) quite trivial. For soundness, let $E1 \neq E2$ be any pair of E-values for which a cheating prover P^* has success, and for which P^* sends the same preimages under h in the two cases. $E1$ must be different from $E2$ in at least one bit position. This implies that there is an H_i in some $H^{(j)}$, which P^* can open and show Hamiltonian, but for which he can also show an appropriate relation to a non-Hamiltonian graph. It is easy to see that this cannot be the case, unless P^* opens at least one commitment to a bit in H_i in two different ways.

Let EE_j be the set of E-values for which P^* successfully produces a proof for a non-Hamiltonian graph with j nodes. If this set constitutes a polynomial fraction of the total number of possibilities, there exists a polynomial time algorithm that (by using rewinding of P^* to the start of step 3 above) generates several elements in EE_j together with the resulting proofs from P^*. By assumption on h, P^* will reveal the same preimages for h in nearly all cases from EE_j, since the preimages must have the same length in order for the proofs to be convincing. Then by the above, P^* will tell us how to open a commitment in two different ways, which contradicts USBCA. Since K is polynomial in k, this means that the union of all $EE_j, j = 1..K$ constitutes a negligible fraction of the total number of cases.

To prove the zero-knowledge property, observe that the simulator can use rewinding of a cheating verifier V^* to extract from the proof in step 1 the trapdoor of the bit commitment scheme. If the simulator manages to find this trapdoor, it can change the contents of any commitment, and the rest of the simulation becomes trivial. Therefore we can simulate all cases perfectly, except those exponentially few ones where V^* manages to cheat in step 1□

Corollary 1 If in the above construction DLS is used for bit commitments, then under CDLA, the resulting protocol is a perfect zero-knowledge argument with preprocessing for GH.

Proof As mentioned before, CDLA implies that collision intractable hash functions exist. Therefore, CDLA alone suffices to implement all the tools we need for the protocol, and soundness and completeness can be proved just as in Theorem 3.

For zero-knowledge, recall that from the data received by the prover for DLS, the prover can check that g generates all of Z_p^*. Therefore, the cases where the verifier cheats successfully in step 1 can still be simulated: the simulator simply finds the discrete log of a by exhaustive search, and then proceeds as above. This situation only occurs with exponentially low probability, and therefore the contribution to the expected running time is only polynomial□

Remark Note that since no message sent in the protocol depends on the upper bound on the theorem size, the fact that such a bound must be known by P is of little consequence: P can generate as many commitments off-line as he wishes, and does not even have to store all the random bits needed for the commitments, because he can generate them pseudorandomly (although the protocol is then only computationally zero-knowledge).

4.3 Perfect Zero-Knowledge Arguments in the Shared String Model

A natural question is of course whether one can construct perfect or statistical zero-knowledge non-interactive arguments in the shared string model, i.e. without preprocessing. For some problems the answer is yes, [BSMP] contains an example for quadratic non-residuosity. No such protocol is known for an NP-complete problem, however. We close this section with a protocol for SAT that partially solves the problem: it is perfect zero-knowledge, but we have not been able to reduce the question of soundness to a generally accepted intractability assumption.

First, recall that it is well-known that there exists a probabilistic polynomial time algorithm A that on input k selects a random k bit prime p and a generator g of Z_p^*, and only fails with negligible probability.

Given the random string α and a satisfiable circuit C of size k, the prover does the following: run A on input k, using the first bits of α as coinflips for A. Let the output be p, g. With the next unused bits, select a constant a uniformly in Z_p^* by repeatedly considering k bit segments of α until one is found that as an integer is less than p. Now encrypt k copies of C as in [BCC] using DLS.

Take the concatenation of all the encryptions thus produced through a one-way function f, and use the first k bits of the output in place of the challenges from the verifier of [BCC]. Include all of the encryptions and answers in the proof.

As for Fiat-Shamir, Schnorr and Guillou-Quisquater signatures (and [BCC] which also mentions the idea of using an f as described above), we can only conjecture the existence of a function f that would make this proof convincing. It is clear, however, that the protocol is perfect zero-knowledge: the simulator can generate p, g as the prover would do it, but choose a with known discrete log base g (and then let the corresponding portion of the shared random string be determined by a).

5 Conclusion and Open Problems

We have shown a non-interactive zero-knowledge proof for SAT, making non-interactive proofs for general NP-statements more direct and efficient.

Based on the assumption that collision intractable hash functions exist, we have shown a non-interactive statistical zero-knowledge argument with preprocessing for GH, in which the preprocessing phase can be made independent of the size of theorem to be proved later.

This protocol works, based on CDLA only, and is then perfect zero-knowledge.

Open problem: find a perfect or statistical zero-knowledge non-interactive argument for an NP-complete problem.

References

[BCC] Brassard, Chaum and Crépeau: "Minimum Disclosure Proofs of Knowledge", JCSS, vol. 37 (1988) pp. 156-189.

[BFM] Blum, Feldman and Micali: "Non-intercative Zero-Knowledge Proof Systems and Applications", Proceedings of STOC 88.

[BKK] Boyar, Krentel and Kurtz: "A Discrete Logarithm Implementation of Perfect Zero-Knowledge Blobs", J. Cryptology, vol 2, no.2.

[CDG] Chaum, Damgård and van de Graaf: "Multiparty Computations Ensuring Privacy of Each Party's Input and Correctness of the Result", Proc. of Crypto 87, Springer Verlag.

[Da] Damgård: "On the Existence of Bit Commitment Schemes and Zero-Knowledge Proofs", Proc. of Crypto 89.

[Da2] Damgård: "A Design Principle for Hash Functions", Proc. of Crypto 89.

[FFS] Fiat, Feige and Shamir: "Zero-Knowledge Proofs of Identity" J. Cryptology, vol 1, pp. 77-94.

[FS] Feige and Shamir: "Zero-Knowledge Proofs of Knowledge in two Rounds", Proc. of Crypto 90.

[FLS] Feige, Lapidot and Shamir: "Multiple Non-Interactive Zero-Knowledge Proof based on a Single Random String", Proc. of FOCS 90.

[Fo] Fortnow: "The Complexity of Perfect Zero-Knowledge", Proc. of STOC 87.

[GM] Goldwasser and Micali: "Probabilistic Encryption", JCSS, vol.28 (1984), pp.270-299.

[GMR] Goldwasser, Micali and Rackoff: "The Knowledge Complexity of Interactive Proof Systems", Proc. of STOC 85.

[KMO] Killian, Micali and Ostrovski: "Minimum Resource Zero-knowledge Proofs", Proc. of Crypto 89.

[LS] Lapidot and Shamir: "Publicly Verifiable Non-Interactive Zero-Knowledge Proofs", Proc. of Crypto 90.

[Na] Naor: "Bit Commitments using Pseudo-Randomness", Proc. of Crypto 89.

[NI] Naor and Impagliazzo: "Efficient Cryptographic Schemes Provavbly as Secure as Subset Sum", Proc. of STOC 89.

[NY1] Naor and Yung: "Universal One-Way Hash Functions and their Cryptographic Applications", Proc. of STOC 90.

[BSMP] Blum, De Santis, Micali and Persiano: "Non-Interactive Zero-Knowledge", SIAM J.Computing, Vol.20, no.6, 1991.

[SMP] De Santis, Micali and Persiano: "Non-Interactive Zero-knowledge with Preprocessing", Proc. of Crypto 88.

[SP] De Santis and Persiano: "Public-Randomness in Public-Key Cryptography", Proc. of EuroCrypt 90.