Senior Research Thesis
Direct Zero-Knowledge Proofs

Jeremiah Blocki
Advisor: Manuel Blum

May 1, 2009

Abstract

Definition A *Zero-Knowledge Proof is an interactive protocol which allows one party (the prover) to prove a statement (S) to another party (the verifier) without revealing anything beyond the truth of S. The protocol allows the prover and verifier to toss private coins.*
If S is true then the prover should always be able to convince the verifier that the statement is true without revealing anything else. For example, Goldreich, Micali, and Wigderson [1991] found a protocol which allowed the prover to convince a verifier that a graph G is k-colorable without revealing anything else. Consequently, all languages in NP are known to have Zero-Knowledge Proofs because they can be reduced to Graph Coloring. However, there are no known *direct* Zero-Knowledge Proofs for many NP-Complete languages. I present direct Zero-Knowledge Proof protocols for Subset Sum, Clique, SAT and Integer Programming.

1 Introduction

1.1 Problem Description

A Zero-Knowledge Proof Protocol must satisfy three properties:

1. **Completeness** - if S is true then the Prover can always convince the Verifier of this by following the protocol.

2. **Soundness** - if S is false then the protocol guarantees that with high probability the Verifier will catch the Prover cheating.

3. **Zero-Knowledge** - The Verifier can simulate an *identical* interaction transcripts without the Prover because he knows the results of his private coin tosses. Informally, we say that the verifier learns nothing more than "S is true." In particular the verifier will not know why S is true and will not be able to prove to anyone that S is true.

1.2 Problem History

Goldreich et al. [1991] gave the first Zero Knowledge Proof scheme for an NP-Complete language, Graph Coloring. Therefore, every language in NP has a Zero-Knowledge Proof scheme by reduction to Graph Coloring. This is nice in theory, but in practice it may be extremely difficult to reduce a problem to Graph Coloring.

SAT was the first language shown to be NP-Complete language. Other NP-Complete have been established by reductions either from SAT or another NP-Complete language. Cook showed that SAT is NP-Complete by considering the Tableau of a Turing Machine. Therefore, the first step in the known reduction from Subset-Sum to Graph Coloring involves building a Turing Machine to verify Subset-Sum solutions. For example here is the Zero-Knowledge Proof protocol for Hamiltonian Cycle:

1. Built a Nondeterministic Turing Machine (M) which solves the Hamiltonian Cycle problem
2. Build a SAT formula ϕ_G based on the computation tableau of $M(G)$
3. Reduce the SAT instance ϕ_G to a Graph Coloring Instance G_C
4. Run the Zero-Knowledge Proof scheme for Graph Coloring on G_C

In theory this is fine, we have provided a Zero-Knowledge Proof Protocol for Hamiltonian Cycle. However, in practice no one would ever build a Nondeterministic Turing Machine to solve Hamiltonian Cycle!

Definition A *Direct* Zero-Knowledge Proof protocol for a language $L \in NP$ is a Zero-Knowledge Proof protocol which does not reduce the language L to some other language L' in any of the steps.

Intuitively, a *Direct* Zero-Knowledge Proof protocol is one a programmer could implement assuming there is some way to commit and hide

information (one-way function, phyical methods etc..). This prevents the prover from changing information during the protocol, but prevents the verifier from seeing the information that the prover commited to maintain Zero-Knowledge.

1.3 Applications

1. Password Authentication Protocols - Suppose that Alice knows the solution to a Subset Sum problem. Because Subset Sum is $NP - hard$ it is reasonable to believe that nobody else will ever find the solution. Now Alice can publish the Subset Sum problem and use her solution as her password. Because she has the password (solution) she can use Zero-Knowledge Proof techniques to convince others that she knows the password without giving away any information about the password.

2. Secure Multiparty Computation [Goldreich et al., 1987] - In many cryptographic protocols it is necessary to enforce honesty and maintain privacy at the same time. These may seem like contradictory goals, but using Zero-Knowledge Proofs it is often possible for a participant to prove that his behavior is honest without revealing private information.

3. Message Authentication Protocols - I will present a simple message authentication protocol based on the Zero-Knowledge Proof protocol for Subset-Sum. This is similar to message signing in RSA, with one key difference. If Alice signs a message, then Bob can later prove to anyone that Alice signed the message. In a message authentication protocol Alice convinces Bob that she is the one sending this message, but later Bob will not be able to convince anyone else that Alice sent him the message.

As we noted before, most Zero-Knolwedge Proof protocols assume that there is some way to commit and hide information. Typically this would be achieved by a One-Way Function or through phyiscal means.

Definition A function $f : \{0,1\}^* \to \{0,1\}^*$ is one-way if there is a polynomial time algorithm A_f which computes f, and for every randomized polynomial time algorithm $A(y,r)$, polynomial $p(n)$ and sufficiently large n

$$\Pr_{x\in\{0,1\}^n, y\in\{0,1\}^{p(n)}}[f(A(f(x),r)) = f(x)] \leq \frac{1}{p(n)}$$

Intuitively, it is a function that is easy to compute, but hard to invert for almost any value of x. If one-way functions do exist then it is easy to prove that $P \neq NP$. Unfortuanatley, for $NP - Complete$ languages it is unlikely that we can remove this need to commit and hide information [Fortnow, 1987, Boppana et al., 1987]. However, Ben-Or et al. [1988] showed that if there are two provers, who are allowed to communicate before, but not during the protocol, the assumption that One-Way Functions exist can be removed entirely.

1.4 Results

In practice, it would be nice to have direct Zero-Knowledge Proof schemes. I will present direct Zero-Knowledge Proof protocols for Subset Sum, Clique, SAT and Integer Programming. All of these problems are *NP*-Complete.

2 Subset Sum

2.1 Problem Definition

Definition A *Subset Sum* instance $I =< S, m >$ consists of a set S of integers and an integer m. I is a yes instance if and only if

$$\exists X \subseteq S, \Sigma_{x \in X} x = m$$

The *Subset Sum* problem is well known to be NP-Complete [Karp, 1972]. For simplicity, I present a protocol for the Equal Partition Problem, a special case of Subset Sum which is also known to be NP-Complete. Given a set $S = \{v_1, ..., v_n\}$ find a disjoint partition $S = S_1 \bigcup S_2$ such that:

1. $||S_1|| = ||S_2||$
2. $S_1 \bigcap S_2 = \emptyset$
3. $\Sigma_{x \in S_1} x = \Sigma_{x \in S_2} x$

2.2 Protocol

Prover: Assume that that the prover knows such set X with

$$||X|| = \frac{n}{2}$$

Define $M = \Sigma_{x \in S} x$

1. Generate $r_1, ..., r_n$, where r_i is chosen uniformly at random from $\{0, ..., M\}$
2. Compute $R_1, ..., R_n$ where $R_i + r_i = v_i \mod M + 1$
3. Commit (but hide)
$$A = \Sigma_{i=1}^{n} r_i b_i$$
and
$$B = \Sigma_{i=1}^{n} R_i b_i$$
where $b_i = 1$ indicates that $v_i \in X$ and $b_i = 0$ indicates $v_i \notin X$.
4. Commit (but hide) a column permuted version of the following table:

v	v_1	v_2 ...
r	r_1	r_2 ...
R	R_1	R_2 ...
$v_i \in X$?	b_1	b_2...

Verifier:
Now the *verifier* can ask to see exactly one of the following:

1. All of the triples (v_i, r_i, R_i) (checking that $R_i + r_i \equiv v_i \mod M + 1$)
2. $R_1, ..., R_n, b_1, ..., b_n$ and A, B (checking that $\Sigma_{i=1}^{n} b_i R_i \equiv B \mod M+1$ and $A + B \equiv k \mod M + 1$)
3. $r_1, ..., r_n, b_1, ..., b_n$ and A, B (checking that $\Sigma_{i=1}^{n} b_i r_i \equiv 0 \mod M + 1$ and $A + B \equiv k \mod M + 1$)

If any check fails then the verifier rejects immediately.

Claim 2.1 *The above protocol is a Zero-Knowledge Proof scheme*

Proof We must show that the protocol satisfies the Zero-Knowledge, Completeness and Soundness conditions.

Zero-Knowledge The key intuition is that r_i by itself is just a random number. Similarly, R_i is just a random number without r_i. Thus, at each step, the verifier is shown numbers which he could have generated randomly. Formally, the verifier could easily simulate choice 1 by himself as follows:

1. Pick $r_1, ..., r_n$ uniformly at random.

2. Pick $R_1, ..., R_n$ such that

$$R_i + r_i \equiv v_i \mod M + 1$$

Similarly, the verifier could simulate choice 3 as follows

1. Pick the $r_1, ..., r_n$ from $\{0, ..., M\}$
2. Pick $b_1, ..., b_n$ values uniformly at random such that

$$||\{i : b_i = 1\}|| = \frac{n}{2}$$

3. Pick A such that

$$\Sigma_{i=1}^{n} b_i r_i \equiv A \mod M + 1$$

4. Pick B such that $A + B \equiv k \mod M + 1$

Finally, the verifier can simulate choice 2 as follows:

1. Pick the $R_1, ..., R_n$ from $\{0, ..., M\}$
2. Pick $b_1, ..., b_n$ values uniformly at random such that

$$||\{i : b_i = 1\}|| = \frac{n}{2}$$

3. Pick B such that

$$\Sigma_{i=1}^{n} b_i R_i \equiv B \mod M + 1$$

4. Pick A such that $A + B \equiv k \mod M + 1$

Completeness If the prover knows of a set X then it is easy to verify there is no way for him to be caught if he follows the protocol.

Soundess Suppose that one of the following was true

1. $\exists i$ s.t

$$R_i + r_i \not\equiv v_1 \mod M + 1$$

2. $\Sigma_{i=1}^{n} b_i R_i \not\equiv B \mod M + 1$
3. $\Sigma_{i=1}^{n} b_i r_i \not\equiv A \mod M + 1$
4. $A + B \not\equiv k \mod M + 1$

5. $|\{i : b_i = 1\}| \neq \frac{n}{2}$

Then a verifier who selects from the three options uniformly at random has at least a $\frac{1}{3}$ chance of catching the prover.

Claim 2.2 *Suppose that all four statements were always false, so that it is impossible for the verifier to ever catch the prover. Then the set*

$$X = \{v_i | b_i = 1\} \subset S$$

is the correct solution.

Proof First, note that $|X| = \frac{n}{2}$ by statement 4.

$$\begin{aligned}
\Sigma_{x \in X} x &\equiv \Sigma_{i=1}^{n} b_i v_i \mod M + 1 && (1)\\
&\equiv \Sigma_{i=1}^{n} b_i (r_i + R_i) \mod M + 1 && (2)\\
&\equiv \Sigma_{i=1}^{n} b_i r_i + \Sigma_{i=1}^{n} b_i R_i \mod M + 1 && (3)\\
&\equiv A + B \mod M + 1 && (4)\\
&\equiv k \mod M + 1 && (5)
\end{aligned}$$

but $\Sigma_{x \in X} x \leq \Sigma_{x \in S} x \leq M$. Therefore,

$$\Sigma_{x \in X} x = k$$

2.3 Example

1. $S = \{59, 32, 23, 44, 60, 85, 90, 60\}$
2. $k = 248$
3. $M = \Sigma_{x \in S} = 453$
4. $X = \{44, 60, 85, 59\}$

2.3.1 Peggy

Peggy knows X and generates the following table, but hides all of the cells of the table from Victor.

v	44	32	60	85	59	23	90	60
r	94	314	103	0	257	387	27	433
R	138	346	163	85	316	410	117	39
$v_i \in X$?	1	0	1	1	1	0	0	0
A and B	A	B	0	0	0	0	0	0

2.3.2 Victor

Victor sees the following table

v	*	*	*	*	*	*	*	*
r	*	*	*	*	*	*	*	*
R	*	*	*	*	*	*	*	*
$v_i \in X$?	*	*	*	*	*	*	*	*
A and B	*	*	*	*	*	*	*	*

Victor now has three choices:

1. Victor asks for proof that $R_i + r_i \equiv v_i \mod M + 1$. In response Peggy reveals this only this part of the table

v	44	32	60	85	59	23	90	60
r	94	314	103	0	257	387	27	433
R	138	346	163	85	316	410	117	39
$v_i \in X$?	*	*	*	*	*	*	*	*
A and B?	*	*	*	*	*	*	*	*

2. Victor asks for proof that $\Sigma_{i=1}^{n} b_i R_i \equiv B \mod M + 1$ and that $A + B \equiv k \mod M + 1$. In response Peggy reveals another part of the table:

v	*	*	*	*	*	*	*	*
r	*	*	*	*	*	*	*	*
R	138	346	163	85	316	410	117	39
$v_i \in X$?	1	0	1	1	1	0	0	0
A and B	A	B	0	0	0	0	0	0

3. Victor asks for proof that $\Sigma_{i=1}^{n} b_i r_i \equiv 0 \mod M + 1$. In response Peggy reveals part of the table

v	*	*	*	*	*	*	*	*
r	94	314	103	0	257	387	27	433
R	*	*	*	*	*	*	*	*
$v_i \in X$?	1	0	1	1	1	0	0	0
A and B	A	B	0	0	0	0	0	0

2.4 Extending the Protocol to regular Subset Sum

The protocol can be adapted to the Subset Sum problem, though there is one major technicality. Our protocol cannot reveal the size ($|X|$) of our Subset Sum solution $X \subset S$. We no longer guarantee that $|X| = \frac{|S|}{2}$. To fix this we create a new set S' by padding the set S with $|S|$ zero entries. Clearly, we cannot generate any new Subset Sums with S' because we just added 0 a bunch of times. However, if there was a solution $X \subset S$ of size $|X| < |S|$, we can create $X' \subset S'$ of size $|S|$ by padding X with zeros. Thus we may assume without loss of generality that our Subset Sum solutions in S' have size $\frac{|S'|}{2}$.

2.5 A Message Authentication Protocol using Subset Sum

Given a set S and a number k suppose that Alice is the only person that knows the solution X to this Subset-Sum problem. Now to authenticate an m-bit message, Alice can define a new set

$$S' = S \bigcup \{2^{\lceil \log_2 M \rceil + 1}, ..., 2^{\lceil \log_2 M \rceil + m}\}$$

Notice that a m-bit message can also be interpreted as subset

$$T \subset \{2^{\lceil \log_2 M \rceil}, ..., 2^{\lceil \log_2 M \rceil + m}\}$$

Now Alice sets $X' = X \bigcup T$ and $k' = \Sigma_{x \in X'} x$. To authenticate the message to Bob, Alice simply proves that she has X' such that

$$\Sigma_{x \in X'} x = k'$$

Notice that it is easy for anyone to compute k' given k and the original message (T), so Alice is never giving away information about X. Also, it is easy for anyone, including Bob, to reconstruct T (and hence Alice's message) from k'. Hence, anyone who had X' would be able to immediately construct X. As long as Alice is the only person who knows X so she will also be the only person who knows X'. Notice that Alice can prove to Bob what message she is sending, but even after Bob is convinced he will be unable to prove to anyone else that Alice sent him this message.

3 Clique

3.1 Problem Definition

Definition A Clique instance $I =< G, k >$ is a undirected graph G and an integer k. I is a yes instance if and only if G contains a clique of size $\geq k$.

Clique was one of Richard Karp's original 21 NP-complete problems [Karp, 1972].
Let M_G denote the adjacency matrix of G.

3.2 Protocol

Prover:

1. Generate an isomorphic graph G'
2. Write down (but hide) $M_{G'}$

Verifier:

1. Ask to see that G' is isomorphic to G (the prover just reveals $M_{G'}$ and the permutation).
2. Ask to see that G' contains a $k-clique$ (the prover reveals which nodes are in the k-clique and reveals the corresponding cells in the adjacenty matrix ...these should all be 1).

Claim 3.1 *The above protocol is a Zero-Knowledge Proof scheme*

Proof We must show that the protocol satisfies the Zero-Knowledge, Completeness and Soundness conditions.

Zero-Knowledge Victor could simulate option 1 by simply generating an isomorphic graph and writing down $M_{G'}$. Victor can simulate option 2 by just picking k nodes at random and writing down some adjaceny matrix there all of the edges between these nodes is 1.

Completeness If the prover knows of a $k-clique$ in the graph then the prover can always write down an isomoprhic graph G' for which he knows a $k-clique$ and never get caught.

Soundess If the prover does not know of a k-clique then either G' is not isomorphic or the prover does not know of a k-clique in G' either. In either case the verifier can catch the cheat with probability $\frac{1}{2}$ by selecting from the two options randomly.

Note: Essentially the same protocol works for many graph problems such as: Hamiltonian Path/Cycle, Subgraph Isomporphism, Independent Set and Vertex Cover

3.3 A More Efficient Zero-Knowledge Proof Protocol for Graph 3-Coloring

Given a graph G with n vertices, a 3-Coloring is a partition of the vertices into three sets V_1, V_2, V_3 such that there are no edges between the sets

$$i \neq j \rightarrow E(V_i, V_j) = \emptyset$$

generate a new graph G' with $3n$ vertices by adding $2n$ vertices (but no new edges). Clearly, G' is 3-Colorable if and only if G is 3-Colorable. Also, note that G' can be partitioned into $|V'_1| = |V'_2| = |V'_3| = n$ such that

$$i \neq j \rightarrow E(V'_i, V'_j) = \emptyset$$

Let M be the adjacency matrix of G'. The prover commits, but hides, a permuted version of M ($\pi(M)$), as well as the permuted sets $C_i = \pi(V'_i)$. The Verifier can ask to see

1. C_1, C_2, C_3 and the parts of $\pi(M)$ which would correspond to possible edges between the sets (expecting all 0 entries)

2. Proof that $\pi(M)$ is a valid permutation of M (expecting to be shown a valid permutation π such that $\pi(M) = M$)

The proof achieves $\frac{1}{2}$ soundess in one round, which is better then the $O(\frac{1}{n})$ soundness achieved by Goldreich et al. [1991].

4 SAT

4.1 Definition of Problem

SAT is the original NP-Complete problem Cook [1971]. For simplicity, I present a Zero-Knowledge Proof for the Exactly One in $3 - SAT$ problem, which is also well known to be NP-Complete Schaefer [1978]. We are given a $3 - SAT$ formula ϕ with variables $x_1, ..., x_n$ and clausese $C_1, ..., C_m$.

$$C_i = \{\ell_{i,1}, \ell_{i,2}, \ell_{i,3}\}$$

4.2 Protocol

Prover:

1. For each variable x_i (replace x_i with $\bar{x}_i$ and replace $\bar{x}_i$ with x_i) with probability $\frac{1}{2}$. Notice that the formula remains equivalent when we do this. Let $N_{x_i} : \{x_i\} \rightarrow \{x_i, \bar{x}_i\}$ denote these choices.

2. Permute the variables and the clauses. Let x'_i and C'_j denote the permuted clauses and variables. Let π_1 denote the permutation of the x_i, and π_2 denote teh permutation of the clauses.

3. Commit (but hide) the new list of clauses $\{\pi_2(C_1), ..., \pi_2(C_2)\}$

4. Commit (but hide) the satisfying assignment to the permuted formula

Verifier: The verifier can ask for

1. A Proof that the new formula is equivalent (the prover can just reveal $\{\pi_2(C_1), ..., \pi_2(C_2)\}$, π_1, π_2, and $N_{x_i}, \forall i$ and the verifier can check that the formula is indeed equivalent).

2. Proof that a particular clause $\pi_2(C_i)$ is satisfied by the hidden assignment (Suppose that $C_i = \{\ell_1, \ell_2, \ell_3\}$, then the prover reveals the assignment for $\pi_1(\ell_1), \pi_1(\ell_2), \pi_1(\ell_3)$, all the other clauses remain hidden as well as the permutaions)

Claim 4.1 *The above protocol is a Zero-Knowledge Proof Protocol for 3-SAT*

Proof We must show that the protocol satisfies the Zero-Knowledge, Completeness and Soundness conditions.

Zero Knowledge: The verifier could simulate this protocol by himself.

1. If he chooses option 1 then generate $\pi_1, \pi_2, N_{x_i} \forall i$ at random (randomly permute the variables and clauses)

2. If he chooses option 2 then generate π_1, π_2 randomly permute the variables and clauses then randomly select a clause $\pi_2(C_i)$, and randomly make up a partial assignment such that the clause is true (in the assignment exactly one of the literals will be true).

Completeness: If Peggy knows of a satisfying assignment, she can never be caught by Victor if she just uses the actual assignment.

Soundness: If the Prover does not know of a satsifying assignment then she can either

1. Write down some other nonequivalent formula (getting caught by option 1)

2. Write down some assignment which doesn't satsify all clauses (getting caught in option 1 with probility at least $\frac{1}{m}$)

With some work the protocol can be extended to regular SAT.

5 Integer Programming

5.1 Introduction to Problem

Without loss of generality linear program is a set of equations of the form

$$a_1x_1 + \ldots + a_nx_n = d$$

where $d, a_1, ..., a_n$ are constants and $x_1, ..., x_n$ are variables. 0-1 Integer Programming is also one of Karp's 21 NP-Complete problems [Karp, 1972]. It adds the constraint $x_i \in \{0, 1\}$. This problem is very similar to the Subset Sum problem (with $S = \{a_1, ..., a_n\}$ and $d = m$) except that we may now have multiple constraints. The Zero-Knowledge Proof techniques are also very similar.

5.2 Protocol

Given constraints

$$a_{1,i}x_1 + \ldots + a_{n,i}x_n = d_i$$

for $i = 1, \ldots, m$. Set $M = \max_i \Sigma_{j=1}^{n} a_{j,i}$.

Prover:

1. Create dummy variables $x_{n+1}, ..., x_{2n}$ such that $x_{n+i} = \bar{x_i}$.

2. Pick $y_1, ..., y_{2n}$ uniformly at random from $\{0, ..., M\}$. For the first row, write down (but hide) $y_1, ..., y_{2n}$.

3. Compute $z_1, ..., z_{2n}$ such that

$$y_i + z_i \equiv x_i$$

   For the second row, write down (but hide) $z_1, ..., z_{2n}$.

4. For the other rows, write down (but hide) $a_{1,i}, ..., a_{n,i}, a_{1,i}, ..., a_{n,i}$

5. (In practice the prover also permute the columns, for ease of anaylsis and explanation we pretend that everything is writen in the natural order)

6. For each equation, compute and write down (but hide)

$$v_{1,i} =< z_1, ..., z_n > \cdot < a_1, ..., a_n > \mod M + 1$$

7. Compute and write down (but hide)

$$v_{2,i} =< y_1, ..., y_n > \cdot < a_1, ..., a_n > \mod M + 1$$

Verifier:

1. Proof that the permutated 0-1 Integer Program is equivalent (the prover just shows the permuation of the columns)

2. Show that $z_i + y_i \equiv \{0, 1\} \mod M + 1$ for all $0 < i \leq 2n$ (in fact for exactly n values of i, $z_i + y_i \equiv 1 \mod M + 1$)

3. Show that

$$v_{1,i} =< z_1, ..., z_n > \cdot < a_{1,i}, ..., a_{n,i} > \mod M + 1$$

   was computed correctly

4. Show that

$$v_{2,i} =< y_1, ..., y_n > \cdot < a_{1,i}, ..., a_{n,i} > \mod M + 1$$

   was computed correctly for each equation

5. Show that

$$v_{1,i} + v_{2,i} \equiv d_i \mod M + 1$$

   for each equation

Note: The technique can be extended to regular Integer Programming where we have constraints of the form $x'_i \in \{0, ..., 2^m\}$. Simply replace x'_i with $x_{i,1}, ..., x_{i,m} \in \{0, 1\}$ to represent each bit of x'_i.

Claim 5.1 *The above protocol is Zero-Knowledge*

Proof We must show that the protocol satisfies the Zero-Knowledge, Completeness and Soundness conditions.

Zero Knowledge: The verifier could simulate this protocol by himself. Notice that y_i, z_i are independently random numbers when viewed separately. So any step where we only see y_i or z_i does not reveal anything. Victor could also generate $v_{1,i}, v_{2,i}$ in the last step by picking all of the y_i values at random to obtain $v_{i,1}$ and then setting $v_{2,1}$ to guarantee that

$$v_{1,i} + v_{2,i} \equiv d_i \mod M+1$$

In fact the step where Victor sees the y_i, z_i values and nothing else is ok. There should be exactly n values of i s.t $y_i + z_i \equiv 1 \mod M+1$ and exactly n values of i s.t $y_i + z_i \equiv 1 \mod M+1$. Victor could have simply picked random y_i values and then picked the z_i values so that both statements are true.

Completeness: Clearly, if Peggy follows the protocol she cannot get caught cheating.

Soundness: Suppose Peggy is trying to cheat. To not be caught she must write down an equivalent $0-1$ Integer Program and pick y_i, z_i values such that:

1. For n values of i
$$z_i + y_i \equiv 1 \mod M+1$$
2. For the other n values of i
$$z_i + y_i \equiv 0 \mod M+1$$
3. For all j,
$$< z_1, ..., z_n > \cdot < a_{1,j}, ..., a_{n,j} > + < y_1, ..., y_n > \cdot < a_{1,j}, ..., a_{n,j} > \equiv d_j \mod M+1$$

But then we can set

$$x_i \equiv z_i + y_i \equiv 1 \mod M+1$$

Noting that $x_i \in \{0, 1\}$ and for all j

$$\Sigma_i x_i a_{i,j} \equiv d_j \mod M+1$$

But

$$\Sigma_i x_i a_{i,j} < M + 1$$

Therefore, for all j

$$\Sigma_i x_i a_{i,j} = d_j$$

So if Peggy cheats she will get caught with reasonable probability.

References

M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 113–131. ACM New York, NY, USA, 1988.

RB Boppana, J. Hastad, and S. Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25(2):127–132, 1987.

S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM New York, NY, USA, 1971.

L. Fortnow. The complexity of perfect zero-knowledge. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 204–209. ACM New York, NY, USA, 1987.

O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM New York, NY, USA, 1987.

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 (JACM)*, 38(3):690–728, 1991.

R.M. Karp. *Reductibility among combinatorial problems*. Univ. of California, 1972.

T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 216–226. ACM New York, NY, USA, 1978.