A Zero-Knowledge Proof for the Hidden Subset Sum Problem

Charles F. de Barros[1]

[1]Departamento de Ciência da Computação (DCOMP)
Universidade Federal de São João Del Rei (UFSJ)
São João Del Rei – MG – Brazil

charlesbarros@ufsj.edu.br

Abstract. *In this paper, we propose a zero-knowledge proof for a special case of the hidden subset sum problem. This problem was presented by [Boyko et al. 1998] as the underlying problem of methods for generating random pairs of the form* $(x, g^x \pmod p)$ *using precomputations. The proof we propose is an adaptation of a zero-knowledge protocol for the subset sum problem presented by [Blocki 2009].*

1. Introduction

Zero-knowledge proofs have interesting cryptographic applications, as they provide means for one party to prove knowledge of some secret, without revealing any information about that secret to the other party. This is a very desirable property in authentication scenarios, for example.

It is known that some conventional authentication protocols, such as password-based ones, may be vulnerable to offline attacks, based on the use of dictionaries or rainbow tables, because hashes of the passwords must be transmitted or stored somewhere. If an adversary has access to these hashes, he may compare them to a list of pre-computed hashes of known passwords. Whenever a match is found, he is able to authenticate to the system.

In this scenario, we say that the adversary *learns* something by intercepting the password hash, as long as he is able to compare it to other hashes. In a zero-knowledge protocol, the adversary learns nothing by eavesdropping on the communication between the two parties.

A simple way of applying this concept to the design of authentication protocols is to provide zero-knowledge proofs for hard mathematical problems. In this case, one party could prove that she knows the solution for a given instance of the problem, without revealing any information about that solution.

In this paper, we propose a zero-knowledge proof for the hidden subset sum problem. The proof we propose is based on a protocol by [Blocki 2009] for the classical subset sum problem. The hidden subset sum problem was presented by [Boyko et al. 1998] as the underlying problem of methods for generating randomly distributed pairs of the form $(x, g^x \pmod p)$ using precomputations. This new mathematical problem was considered potentially as hard, or even harder than the classical subset sum problem, since no algorithm was known to solve it, apart from exhaustive search.

The hardness of the problem was thoroughly analysed by [Nguyen and Stern 1999], which established a security criterion for using the generators based on hidden subset sums. They proposed a lattice-based attack, which was likely to succeed whenever the parameters of the problem satisfied certain conditions.

Nonetheless, no further cryptographic applications for the hidden subset sum problem were proposed, maybe due to the lack of a more rigorous hardness result. In this paper, we shed some light on the subject by presenting a possible relation between the hardness of the hidden subset sum problem and the hardness of the classical subset sum problem.

1.1. Roadmap

This paper is organized as follows: in Section 2, we fix some notations and present some theoretical review on subset sums, hidden subset sums and interactive zero-knowledge proofs. In Section 3, we review Blocki's zero-knowledge proof for the subset sum problem, and in Section 4 we extend Blocki's protocol to the hidden subset sum problem. A brief performance analysis is presented in Section 5, and in Section 6 we present our final remarks and proposals for future work.

2. Theoretical Review

In this section, we provide a brief review on subset sums, hidden subset sums and zero-knowledge proofs. From now on, we denote by $\mathcal{I}_n$ the set of indices $\{1, \cdots, n\}$, and the cardinality of a set $\mathcal{A}$ shall be denoted by $|\mathcal{A}|$. We also use the symbol $\oplus$ to denote the xor operation (sum modulo 2).

2.1. The Subset Sum Problem

The subset sum problem is well known in complexity theory. Essentially, it consists of, given a set

$$\mathcal{A} = \{a_1, \cdots, a_n\} \tag{1}$$

of integers and a *target* integer t, deciding whether exists a subset of $\mathcal{A}$ whose sum is t. The problem is known to be NP-complete [Cormen et al. 2001].

We may think of the problem of deciding whether exists a subset of indices $\mathcal{S} \subset \mathcal{I}_n$ such that

$$\sum_{j \in \mathcal{S}} a_j = t. \tag{2}$$

An interesting variant of the subset sum problem consists of finding two subsets of the same size with the same sum. This variant is known as the *equal partition problem*.

Problem 2.1 (Equal Partition Problem) *Given a set $\mathcal{A} = \{a_1, \cdots, a_n\} \subset \mathbb{Z}$, find a partition $\mathcal{I}_n = \mathcal{S}_1 \cup \mathcal{S}_2$ such that*

1. $|\mathcal{S}_1| = |\mathcal{S}_2| = n/2$;
2. $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$; *and*
3. $\sum_{j \in \mathcal{S}_1} a_j = \sum_{j \in \mathcal{S}_2} a_j = t$ *for some target integer* t.

Note that the problem can also be formulated over the ring of integers modulo M, for some integer M. In fact, given M, n positive integers, $t \in \mathbb{Z}_M$ and a set

$$\mathcal{A} = \{a_1, \cdots, a_n\} \subset Z_M, \tag{3}$$

decide whether exists $\mathcal{S} \subset \mathcal{I}_n$ such that

$$\sum_{j \in \mathcal{S}} a_j \equiv t \pmod{M}. \tag{4}$$

In its non-decisional version, the goal of the subset sum problem is to find the subset $\mathcal{S}$, if such exists. Note that both the equal partition and the modular variants are still NP-complete. In fact, we can reduce any instance of subset sum to an instance of equal partition by simply appending n zeros to the input set (considered here as having cardinality n). On the other hand, any instance $\mathcal{A} = \{a_1, \cdots, a_n\}$ of the subset sum problem over the integers is also an instance over $\mathbb{Z}_{M+1}$, where $M = \sum_{j=1}^{n} a_j$.

Despite the fact that the problem is NP-complete, there are methods to solve "easy" instances, such as superincreasing sequences and low-density subset sums. We focus our attention on the latter. We define the *density* in (1) as

$$d = \frac{n}{\log_2(\max\limits_{1 \leq i \leq n} a_i)}. \tag{5}$$

Similarly, in (3), the density is given by

$$d = \frac{n}{\log_2(M)}. \tag{6}$$

It was shown by [Lagarias and Odlyzko 1985] that subset sums with sufficiently low density, namely less than $0.6463\cdots$ can be efficiently solved by using lattice reduction methods, such as [Lenstra et al. 1982] and [Schnorr 1994]. Posteriorly, [Coster et al. 1992] improved the method to solve subset sums with density less than $0.9408\cdots$.

The subset sum problem is a special case of the knapsack problem, which underlies some well known public-key cryptosystems, such as [Merkle and Hellman 1978], [Chor and Rivest 1988] and [T. Okamoto and Uchiyama 2000]. The core idea of a knapsack-based cryptosystem is to use an "easy" knapsack as the secret key (a superincreasing one, for example), while the public key is a "hard" knapsack. However, these cryptosystems may be prone to low-density attacks, which exploit the fact that the public key is a knapsack with low density.

2.2. The Hidden Subset Sum Problem

The hidden subset sum problem, as the name suggests, was inspired by the classical subset sum problem. It was presented by [Boyko et al. 1998] as the underlying problem of methods for generating random pairs of the form $(x, g^x \pmod{p})$ using precomputation. This is useful for protocols based on the discrete-logarithm problem, such as El Gamal [El Gamal 1985] and DSS [NIST 1994] signatures, where generation of the aforementioned pairs is considerably expensive.

We present here a brief description of one of the generators based on the hidden subset sum problem. Keep in mind that the goal is to generate randomly distributed pairs of the form $(x, g^x \pmod p)$. Firstly, choose a prime number p and $g \in \mathbb{Z}_p^*$ of order M. Generate random integers $a_1, \cdots, a_n \in \mathbb{Z}_M$ and compute $\beta_i = g^{a_i}$, for $i = 1, \cdots, n$. Store the $a_i's$ and the $\beta_i's$ in a table.

Whenever a pair $(x, g^x \pmod p)$ is needed, choose a random subset $S \subset \mathcal{I}_n$ and compute

$$b = \sum_{i \in S} a_i \pmod M. \tag{7}$$

If $b = 0$, stop and start again with a new subset S. As soon as $b \neq 0$, compute

$$B = \prod_{i \in S} \beta_i \pmod p. \tag{8}$$

The value of x is given by b and g^x is given by B. Output the pair (b, B), where we have clearly that $B = g^b \pmod p$. In an active attack, an adversary may gather many values of b, say $b_1, \cdots, b_m$, and try to figure out the values of the a_i's. This is an instance of the hidden subset sum problem, which we state below.

Problem 2.2 (Hidden Subset Sum Problem) *Let n, m, M be positive integers. We are given $\mathcal{B} = \{b_1, \cdots, b_m\} \subset \mathbb{Z}_M$, and the goal is to find $\mathcal{A} = \{a_1, \cdots, a_n\} \subset \mathbb{Z}_M$ such that, for all $i = 1, \cdots, m$, there exists $\mathcal{S}_i \subset \mathcal{I}_n$ such that*

$$b_i \equiv \sum_{j \in \mathcal{S}_i} a_j \pmod M. \tag{9}$$

Until the publication of the work by [Nguyen and Stern 1999], no algorithm to solve this problem was known. In their paper, Nguyen and Stern present a low-density attack on the hidden subset sum problem, exploiting the low density of the hidden sequence $a_1, \cdots, a_n$. After running a series of experiments, they conclude that, for the case where $d = 1$, the attack is already very unlikely to succeed, even if sparse subset sums are chosen.

We remark that, although there are no formal hardness results for the hidden subset sum problem, we have strong reasons to believe that it is at least as hard as the classical subset sum problem.

In order to throw some light on this discussion, we must precisely define what it means to solve an instance of the hidden subset sum problem. If by solving we mean simply outputting the set $\mathcal{A}$, then we are left with the problem of checking the validity of the solution, since we would need to solve m instances of the subset sum problem. Hence, we consider that an algorithm solves an instance of the hidden subset sum problem if it outputs the set $\mathcal{A}$ *and* the subsets $\mathcal{S}_i$, for $i = 1, \cdots, m$, so that the decisional version of the problem can be considered as a member of NP.

Provided with a precise notion of what it means to solve the hidden subset sum problem, we may state the following conjecture regarding its hardness:

Conjecture 2.1 *Let $\mathcal{O}$ be an oracle that takes as input three integers M, m, n and a set $\mathcal{B} = \{b_1, \cdots, b_m\} \subset \mathbb{Z}_M$, and outputs $\mathcal{A} = \{a_1, \cdots, a_n\} \in \mathbb{Z}_M$, together with the*

subsets $\mathcal{S}_i \subset \mathcal{I}_n$ such that

$$b_i \equiv \sum_{j \in \mathcal{S}_i} a_j \pmod{M} \tag{10}$$

for $i = 1, \cdots, m$. Given $\mathcal{A} = \{a_1, \cdots, a_n\} \in \mathbb{Z}_M$ and an integer $t \in \mathbb{Z}_M$, can we find a subset $\mathcal{S} \subset \mathcal{I}_n$ such that

$$t \equiv \sum_{j \in \mathcal{S}} a_j \pmod{M}, \tag{11}$$

given access to $\mathcal{O}$?

We sketch a possible proof that the answer is yes, although there are some loose ends that need tying up. We build an instance $\mathcal{B} = \{b_1, \cdots, b_m\}$ of the hidden subset sum problem, where $b_1 = t$ and each b_i, for $i > 1$, is a randomly chosen subset sum of $\mathcal{A}$. Clearly, *there is a probability* that the oracle finds the desired solution, whenever $\mathcal{A}, t$ is a "yes" instance.

In fact, if $\mathcal{A}, t$ is a "yes" instance, then $\mathcal{A}$ is a solution for the hidden subset sum instance, which also contains the subset whose sum is t. Hence, we can solve a subset sum by solving a hidden subset sum. However, this may not be the only solution, since there are no guarantees as for the uniqueness of the solution for a hidden subset sum instance. For example, the instance $\mathcal{B} = \{0, 2, 5, 7\}$ has at least two solutions in $\mathbb{Z}_8$, namely $\{2, 3, 5, 7\}$ and $\{1, 2, 4, 6\}$.

The question remains open as for the number of solutions for a given instance of the hidden subset sum problem. If this number is polynomially bounded, and if we consider an oracle capable of finding all the possible solutions, then the answer for the conjecture above is "yes", and the hidden subset sum problem is at least as hard as the classical subset sum.

In this paper, we propose a variant of the hidden subset sum problem, in which every b_i is the sum of exactly $n/2$ elements of $\mathcal{A}$. We refer to this variant as the *balanced hidden subset sum problem*.

Problem 2.3 (Balanced Hidden Subset Sum Problem) *Let n, m, M be positive integers and a set $\mathcal{B} = \{b_1, \cdots, b_m\} \subset \mathbb{Z}_M$. The goal is to find $\mathcal{A} = \{a_1, \cdots, a_n\} \subset \mathbb{Z}_M$ such that, for all $i = 1, \cdots, m$, there exists $\mathcal{S}_i \subset \mathcal{I}_n$ such that $|\mathcal{S}_i| = n/2$ and*

$$b_i \equiv \sum_{j \in \mathcal{S}_i} a_j \pmod{M}. \tag{12}$$

For instance, if $M = 16$, $n = 4$, $m = 4$ and $\mathcal{B} = \{9, 6, 11, 4\}$, the set $\mathcal{A} = \{2, 7, 4, 13\}$ is a solution, with $\mathcal{S}_1 = \{1, 2\}$, $\mathcal{S}_2 = \{1, 3\}$, $\mathcal{S}_3 = \{2, 3\}$ and $\mathcal{S}_4 = \{2, 4\}$.

As for the hardness of this variant, we may consider a procedure, which is similar to what we have done before, in order to solve the equal partition problem, given access to an oracle that solves the balanced variant of the hidden subset sum problem. Therefore, we assume that for sufficiently high density, namely $d \geq 1$, the balanced variant is hard.

2.3. Interactive Zero-Knowledge Proofs

Zero-knowledge proofs were introduced by [Goldwasser et al. 1985]. In a zero-knowledge proof, also referred to as zero-knowledge protocol, a party called the Prover,

denoted by P, proves to another party (the Verifier, denoted by V) that a given statement S is true, without conveying any information apart from the fact that S is true. In this sense, V gains no *knowledge* from the proof, except for the trueness of the statement.

In a formal sense, this **zero-knowledge property** implies that V could *simulate* his interaction with P, and this simulation would be indistinguishable from a real interaction. This gives us the intuition that the information exchanged between P and V during the protocol must look random. In fact, if everything that V receives from P appears to be random, he gains no extra knowledge, because random data convey no information. Hence, V is able to generate by himself everything he sees from his interaction with P.

Besides the aforementioned property, zero-knowledge proofs must be **complete** and **sound**. Completeness means that, if P is honest and both P and V follow the protocol, then V will accept the proof with probability 1. The protocol is sound if any dishonest prover P' is caught with high probability, provided that the protocol is repeated a sufficiently large number of times. The reader may consult [Quisquater and Berson 1990] for a very interesting illustration of a zero-knowledge proof.

As already mentioned, zero-knowledge proofs may be applied to authentication protocols. In an authentication system, one party must prove its identity to a second party, usually by showing that it knows some secret information (a password, for instance). The problem with this scenario is that, in order to prove knowledge of some secret, typically one must convey information about this secret. For example, one party computes the hash of its secret password and sends it to the second, which gives an adversary the opportunity of launching, for example, a dictionary-based attack against the system.

Zero-knowledge proofs offer the advantage of conveying no information at all about the secret of which one wishes to prove knowledge. Another practical scenario of application includes the proof that a given public key is good (satisfies all the security criteria), without revealing the secret key. For example, proving that an RSA key is the product of two safe primes.

3. A Zero-Knowledge Proof for the Subset Sum Problem

In this section, we review Blocki's zero-knowledge proof for the subset sum problem [Blocki 2009]. In his thesis, Blocki presents a protocol for the equal partition variant, by which the prover P is able to convince the verifier V that he knows a solution for the problem, without conveying any information about this solution.

Let $\mathcal{A} = \{a_1, \cdots, a_n\}$ be the input set, and t the target integer. Remind that the prover wishes to convince a verifier that he knows a set $\mathcal{S} \subset \mathcal{I}_n$ such that $|\mathcal{S}| = n/2$ and

$$\sum_{j \in \mathcal{S}} a_j = \sum_{j \in \mathcal{I}_n \setminus \mathcal{S}} a_j = t. \tag{13}$$

Defining

$$M = \sum_{i=1}^{n} a_i, \tag{14}$$

the protocol works as follows:

1. The prover generates $\beta_1, \cdots, \beta_n \in \mathbb{Z}_{M+1}$ uniformly at random and computes $\gamma_1, \cdots, \gamma_n \in \mathbb{Z}_{M+1}$ such that
$$a_i \equiv \beta_{\sigma(i)} + \gamma_{\sigma(i)} \pmod{M+1}. \tag{15}$$
for $i = 1, \cdots, n$, where σ is a random permutation, which P does not reveal.
2. The prover computes the partial sums
$$b = \sum_{j \in \mathcal{S}_\sigma} \beta_j \tag{16}$$
and
$$c = \sum_{j \in \mathcal{S}_\sigma} \gamma_j, \tag{17}$$
where
$$\mathcal{S}_\sigma = \{\sigma(i) \mid i \in \mathcal{S}\}. \tag{18}$$
3. In the next step, the verifier asks to see exactly one of the following:
   (a) All of the triples $(a_i, \beta_{\sigma(i)}, \gamma_{\sigma(i)})$, checking whether
   $$a_i \equiv \beta_{\sigma(i)} + \gamma_{\sigma(i)} \pmod{M+1}; \tag{19}$$
   (b) $\beta_1, \cdots, \beta_n, b, c$ and $\mathcal{S}_\sigma$, checking whether
   $$\sum_{j \in \mathcal{S}_\sigma} \beta_j \equiv b \pmod{M+1} \tag{20}$$
   and
   $$b + c \equiv t \pmod{M+1}; \tag{21}$$
   (c) $\gamma_1, \cdots, \gamma_n, b, c$ and $\mathcal{S}_\sigma$, checking whether
   $$\sum_{j \in \mathcal{S}_\sigma} \gamma_j \equiv c \pmod{M+1} \tag{22}$$
   and
   $$b + c \equiv t \pmod{M+1}. \tag{23}$$
4. If any of the checks fails, the verifier rejects the proof.

It is easy to see that the protocol is complete. In fact, any prover who knows the subset $\mathcal{S}$ is able to correctly build the subset $\mathcal{S}_\sigma$ and, by following the protocol, he always convinces the verifier, because

$$b + c = \sum_{j \in \mathcal{S}_\sigma} \beta_j + \sum_{j \in \mathcal{S}_\sigma} \gamma_j = \sum_{j \in \mathcal{S}_\sigma} (\beta_j + \gamma_j) = \sum_{i \in \mathcal{S}} (\beta_{\sigma(i)} + \gamma_{\sigma(i)}) = \sum_{i \in \mathcal{S}} a_i = t. \tag{24}$$

The soundness of the protocol comes from the fact that, if a dishonest prover P' is cheating, then he does not know the subset $\mathcal{S}$, which means that he is not able of choosing the proper subset $\mathcal{S}_\sigma$. Instead, he chooses a random subset $\mathcal{S}'$ (which is the wrong subset with overwhelming probability), which leaves him with a probability at least $1/3$ of being caught (at least one of the checkings will fail if the wrong subset was chosen).

Finally, the protocol also has the zero-knowledge property, because each β_i is just a random number, while each γ_i is also a random number without β_i. Thus, the verifier is able to generate by himself everything that the prover shows him at each step of the protocol. Then, he is able to simulate any of the choices (a), (b) and (c) as follows:

1. In order to simulate choice (a), he simply generates $\beta_1, \cdots, \beta_n$ at random and computes $\gamma_1, \cdots, \gamma_n$ such that $\beta_i + \gamma_i \equiv a_i \pmod{M+1}$;
2. In order to simulate choice (b), he generates $\beta_1, \cdots, \beta_n$ uniformly at random, chooses a random subset $\mathcal{S}' \subset \mathcal{I}_n$ such that $|\mathcal{S}'| = n/2$, computes

$$b = \sum_{j \in \mathcal{S}'} \beta_j \tag{25}$$

   and picks c such that $b + c \equiv k \pmod{M+1}$.
3. Analogously, in order to simulate choice (c), he generates $\gamma_1, \cdots, \gamma_n$ uniformly at random, chooses a random subset $\mathcal{S}' \subset \mathcal{I}_n$ such that $|\mathcal{S}'| = n/2$, computes

$$c = \sum_{j \in \mathcal{S}'} \gamma_j \tag{26}$$

   and picks b such that $b + c \equiv k \pmod{M+1}$.

4. Extending the Idea to the Hidden Subset Sum Problem

In this section, we extend Blocki's protocol and present a novel zero-knowledge proof for the hidden subset sum problem. Let us denote the number of combinations of x elements taken y at a time by $\binom{x}{y}$. From now on, let m, n, M be positive integers such that

$$n! \geq \binom{M}{n} \tag{27}$$

and

$$n \geq \log_2 M. \tag{28}$$

The previous inequality implies that the hidden subset has density equal to or greater than 1, in order to mitigate low-density attacks. Furthermore, assume that

$$\left(\frac{m+2}{2m}\right)^m \tag{29}$$

is a negligible amount. A set $\mathcal{B} = \{b_1, \cdots, b_m\} \subset \mathbb{Z}_M$ is given, and P secretly knows another set $\mathcal{A} = \{a_1, \cdots, a_n\} \subset \mathbb{Z}_M$, of which he wishes to prove knowledge, such that each b_i is a subset sum modulo M of exactly $n/2$ elements of $\mathcal{A}$. The protocol works as follows:

1. V randomly chooses $\mathcal{A}_0 = \left\{a_1^{(0)}, \cdots, a_n^{(0)}\right\} \subset \mathbb{Z}_M$ from the uniform distribution and sends it to P.
2. P reveals the set

$$\mathcal{A}_1 = \left\{a_1^{(1)}, \cdots, a_n^{(1)}\right\} \subset \mathbb{Z}_M \tag{30}$$

   such that, for all $i = 1, \cdots, n$,

$$a_i \equiv a_i^{(0)} + a_{\sigma(i)}^{(1)} \pmod{M}, \tag{31}$$

   where σ is a random permutation which P keeps secret.

3. P computes the subsets $\mathcal{S}_i^{(0)}, \mathcal{S}_i^{(1)} \subset \mathcal{I}_n$, satisfying $|\mathcal{S}_i^{(0)}| = |\mathcal{S}_i^{(1)}| = n/2$ for all $i = 1, \cdots, m$, and computes

$$b_i^{(0)} \equiv \sum_{j \in \mathcal{S}_i^{(0)}} a_j^{(0)} \pmod{M} \tag{32}$$

and

$$b_i^{(1)} \equiv \sum_{j \in \mathcal{S}_i^{(1)}} a_j^{(1)} \pmod{M}, \tag{33}$$

where

$$b_i \equiv b_i^{(0)} + b_i^{(1)} \pmod{M} \tag{34}$$

for all $i = 1, \cdots, m$.
4. P reveals the pairs $(b_i^{(0)}, b_i^{(1)})$, for $i = 1, \cdots, m$. If (34) does not hold, V immediately rejects.
5. V sends to P a random bit $w \in \{0, 1\}$ and a random integer $1 \leq k \leq m$.
6. P reveals $\mathcal{S}_k^{(w)}$.
7. V accepts if, and only if

$$\sum_{j \in \mathcal{S}_k^{(w)}} a_j^{(w)} \equiv b_k^{(w)} \pmod{M} \tag{35}$$

and $|\mathcal{S}_k^{(w)}| = n/2$.

The protocol is repeated at least m times. Now we prove that the presented protocol has the properties of completeness, soundness and zero-knowledge.

Theorem 4.1 (Completeness) *The protocol is complete.*

Proof: in fact, let us assume that P is an honest prover, who actually knows the set $\mathcal{A}$. Hence, for each $i = 1, \cdots, m$, he knows $\mathcal{S}_i \subset \mathcal{I}_n$ such that

$$\sum_{j \in \mathcal{S}_i} a_j \equiv b_i \pmod{M}. \tag{36}$$

Hence, for $i = 1, \cdots, m$, he may build the subsets $\mathcal{S}_i^{(0)}$ and $\mathcal{S}_i^{(1)}$ as follows:

$$\mathcal{S}_i^{(0)} = \mathcal{S}_i \tag{37}$$

and

$$\mathcal{S}_i^{(1)} = \{\sigma(j) \in \mathcal{I}_n | j \in \mathcal{S}_i\}. \tag{38}$$

Thus, P is able to correctly compute the pairs $(b_i^{(0)}, b_i^{(1)})$ and, in fact,

$$b_i^{(0)} + b_i^{(1)} = \sum_{j \in \mathcal{S}_i^{(0)}} a_j^{(0)} + \sum_{j \in \mathcal{S}_i^{(1)}} a_j^{(1)} = \sum_{j \in \mathcal{S}_i} a_j^{(0)} + \sum_{j \in \mathcal{S}_i} a_{\sigma(j)}^{(1)}. \tag{39}$$

Since both sums are over the same index, we may write

$$b_i^{(0)} + b_i^{(1)} = \sum_{j \in \mathcal{S}_i} (a_j^{(0)} + a_{\sigma(j)}^{(1)}) = \sum_{j \in \mathcal{S}_i} a_j \equiv b_i \pmod{M}. \tag{40}$$

Therefore, V always accepts the proof, provided that P is honest and both follow the protocol. □

Theorem 4.2 (Soundness) *The protocol is sound.*

Proof: we must show that a dishonest prover P' is caught with high probability. In fact, after receiving the set $\mathcal{A}_0$, a smart cheater could choose m subsets $\mathcal{S}_i^{(0)}$ and compute

$$b_i^{(0)} \equiv \sum_{j \in \mathcal{S}_i^{(0)}} a_j^{(0)}, \tag{41}$$

for $i = 1, \cdots, m$. At this point, the integers $b_i^{(1)}$ are already determined, but the cheater may conveniently choose $n/2$ integers whose sum is, say, $b_1^{(1)}$, and other $n/2$ integers whose sum is $b_2^{(1)}$. It determines the set $\mathcal{A}_1$ along with the subsets $\mathcal{S}_1^{(1)}$ and $\mathcal{S}_2^{(1)}$. In order to determine the subsets $\mathcal{S}_i^{(1)}$ for $i > 2$, he must solve $m - 2$ instances of the subset sum problem over $\mathcal{A}_1$, hence we assume that he is not capable of doing so. Nevertheless, at each iteration, the dishonest prover has two possibilities of being successful at cheating:

1. the verifier chooses the bit 0 at step 5, which happens with probability $1/2$;
2. the verifier chooses the bit 1 and one of the integers 1 or 2 at step 5, which happens with probability $1/m$,

which gives him a probability $1/2 + 1/m$ of successfully cheating. In order to deceive the verifier, the cheater must be successful in all iterations, which will happen with probability at most

$$\left(\frac{m+2}{2m}\right)^m. \tag{42}$$

Since we are assuming that this is a negligible amount, the cheating prover is caught with high probability. □

Theorem 4.3 (Zero Knowledge) *The protocol is zero knowledge.*

Proof: intuitively, everything that V sees looks random. Hence, he is able to generate by himself everything that he receives from P, which means that he may simulate his interaction with P as follows:

1. He chooses $\mathcal{A}_0 = \left\{a_1^{(0)}, \cdots, a_n^{(0)}\right\} \subset \mathbb{Z}_M$ from the uniform distribution.
2. The set $\mathcal{A}_1$ is just a random set, because $\mathcal{A}$ and the permutation σ are unknown. Thus, V simply chooses another random subset

$$\mathcal{A}_1 = \left\{a_1^{(1)}, \cdots, a_n^{(1)}\right\} \subset \mathbb{Z}_M. \tag{43}$$

3. V generates a random string $w \in \{0,1\}^m$, chooses a random subset $\mathcal{S}_i \subset \mathcal{I}_n$ with exactly $n/2$ elements, for all $i = 1, \cdots, m$, and computes

$$b_i^{(w_i)} \equiv \sum_{j \in \mathcal{S}_i} a_j^{(w_i)} \pmod{M}. \tag{44}$$

   Next, he computes $b_i^{(w_i \oplus 1)}$ such that

$$b_i \equiv b_i^{(0)} + b_i^{(1)} \pmod{M}. \tag{45}$$

4. V has simulated the pairs $(b_i^{(0)}, b_i^{(1)})$, for $i = 1, \cdots, m$.

5. V chooses a random integer $1 \leq k \leq m$ and picks the bit w_k from the random string generated at step 3.
6. V simulates the disclosure of the subset $\mathcal{S}_k$.
7. In fact, by construction, we have

$$\sum_{j \in \mathcal{S}_k} a_j^{(w_k)} \equiv b_k^{(w_k)} \pmod{M} \tag{46}$$

and $|\mathcal{S}_k| = n/2$.

Note that, after his interaction with P, V knows that each element of $\mathcal{A}$ is the sum of an element of $\mathcal{A}_0$ with some element of $\mathcal{A}_1$. However, because he does not know the permutation σ chosen by P, the desired solution is among $n!$ possibilities. Assuming that (27) holds, this knowledge provides him with no better method than brute force to find the solution. Thus, in practice, the protocol conveys no extra knowledge for the verifier.

5. Security and Performance Analysis

We have seen that, at each round of the protocol, a dishonest prover successfully cheats with probability $1/2 + 1/m$. Hence, in order to achieve acceptable levels of security, the number of iterations must be sufficiently large. For instance, if $m = 256$, then after 256 rounds a dishonest prover can be caught with probability $1 - 2^{-253}$.

As for the memory space requirements, the prover must store $n(m + \log M)$ bits of secret information, corresponding to the set $\mathcal{A}$, which can be represented with $n \log M$ bits, and the subsets $\mathcal{S}_i$ for $i = 1, \cdots, m$, which can be represented as m strings of n bits. The only public information is the set $\mathcal{B}$, which requires $m \log M$ bits.

During a single round of the protocol, $2\log M(m+n)+n+33$ bits are transmitted (the subsets $\mathcal{A}_0$ and $\mathcal{A}_1$, the pairs $(b_i^{(0)}, b_i^{(1)})$ the bit w and the number k, interpreted as a 32-bit integer, and the subset $\mathcal{S}_k^{(w)}$). The prover must also temporarily store $2mn$ bits corresponding to the subsets $S_i^{(0)}$ and $S_i^{(1)}$, for $i = 1, \cdots, m$.

The parameters $M = 1024$, $n = 256$ and $m = 256$ yield roughly 8.5kB of memory space to store the secret key, 16kB to store temporary information and 1.3kB of transmitted data per round.

6. Final Remarks and Conclusions

We presented a zero-knowledge proof for a special case of the hidden subset sum problem, which can be applied to cryptographic protocols such as authentication systems. The protocol is an adaptation of a zero-knowledge proof for the subset sum problem. The proposal is still not optimal in terms of key sizes and memory requirements, and further research is highly encouraged. Possible optimizations, based on compact representations and variants which require fewer rounds, are part of ongoing research.

Acknowledgements

We would like to thank the anonymous referees for their helpful comments and suggestions.

References

Blocki, J. (2009). Direct zero-knowledge proofs. Senior Research Thesis, B.S. in Computer Science, Carnegie Mellon University.

Boyko, V., Peinado, M., and Venkatesan, R. (1998). Speeding up discrete log and factoring based schemes via precomputations. In *Advances in Cryptology - Eurocrypt '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 221–235.

Chor, B. and Rivest, R. L. (1988). A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Transactions on Information Theory*, 34(5):901–909.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2nd edition.

Coster, M. J., Joux, A., LaMacchia, B. A., Odlyzko, A. M., Schnorr, C.-P., and Stern, J. (1992). Improved low-density subset sum algorithms. *Computational Complexity*, 2(2):111–128.

El Gamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in Cryptology*, pages 10–18, New York, NY, USA. Springer-Verlag New York, Inc.

Goldwasser, S., Micali, S., and Rackoff, C. (1985). The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 291–304, New York, NY, USA. ACM.

Lagarias, J. C. and Odlyzko, A. M. (1985). Solving low-density subset sum problems. *J. Assoc. Comp. Mach.*, 32(1):229–246.

Lenstra, A. K., Lenstra Jr, H. W., and Lovász, L. (1982). Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534.

Merkle, R. C. and Hellman, M. E. (1978). Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 24:525–534.

Nguyen, P. and Stern, J. (1999). *The Hardness of the Hidden Subset Sum Problem and Its Cryptographic Implications*, pages 31–46. Springer Berlin Heidelberg, Berlin, Heidelberg.

NIST (1994). FIPS publication 186: Digital signature standard.

Quisquater, Jean-Jacques, G. L. C. and Berson, T. A. (1990). How to explain zero-knowledge protocols to your children. In *Advances in Cryptology*, CRYPTO '89: Proceedings, pages 628–631. ACM.

Schnorr, C. P. (1994). Block reduced lattice bases and successive minima. *Combinatorics, Probability and Computing*, 3:507–522.

T. Okamoto, K. T. and Uchiyama, S. (2000). Quantum public-key cryptosystems. In *Advances in Cryptology: Proceedings of CRYPTO 2000 (M. Bellare, ed.)*, Lecture Notes in Computer Science, pages 147–165, New York. Springer-Verlag.

