Zero-Knowledge Protocols for the Subset Sum Problem from MPC-in-the-Head with Rejection

Thibauld Feneuil[1,2], Jules Maire[3], Matthieu Rivain[1], and Damien Vergnaud[3,4]

[1] CryptoExperts, Paris, France
[2] Sorbonne Université, CNRS, INRIA, Institut de Mathématiques de Jussieu-Paris Rive Gauche, Ouragan, Paris, France
[3] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
[4] Institut Universitaire de France

Abstract. We propose (honest verifier) zero-knowledge arguments for the modular *subset sum problem*. Previous combinatorial approaches, notably one due to Shamir, yield arguments with cubic communication complexity (in the security parameter). More recent methods, based on the *MPC-in-the-head* technique, also produce arguments with cubic communication complexity.

We improve this approach by using a secret-sharing over small integers (rather than modulo q) to reduce the size of the arguments and remove the prime modulus restriction. Since this sharing may reveal information on the secret subset, we introduce the idea of *rejection* to the MPC-in-the-head paradigm. Special care has to be taken to balance completeness and soundness and preserve zero-knowledge of our arguments. We combine this idea with two techniques to prove that the secret vector (which selects the subset) is well made of binary coordinates.

Our new protocols achieve an asymptotic improvement by producing arguments of quadratic size. This improvement is also practical: for a 256-bit modulus q, the best variant of our protocols yields 13KB arguments while previous proposals gave 1180KB arguments, for the best general protocol, and 122KB, for the best protocol restricted to prime modulus. Our techniques can also be applied to vectorial variants of the subset sum problem and in particular the *inhomogeneous short integer solution* (ISIS) problem for which they provide an efficient alternative to state-of-the-art protocols when the underlying ring is not small and NTT-friendly. We also show the application of our protocol to build efficient zero-knowledge arguments of plaintext and/or key knowledge in the context of *fully-homomorphic encryption*. When applied to the TFHE scheme, the obtained arguments are more than 20 times smaller than those obtained with previous protocols. Eventually, we use our technique to construct an efficient digital signature scheme based on a pseudo-random function due to Boneh, Halevi, and Howgrave-Graham.

1 Introduction

The *(modular) subset sum* problem is to find, given integers $w_1, \ldots, w_n$, t and q, a subset of the w_i's that sum to t modulo q, i.e. to find bits $x_1, \ldots, x_n \in \{0, 1\}$

such that

$$\sum_{i=1}^{n} x_i w_i = t \bmod q. \tag{1}$$

It was shown to be NP-complete (in its natural decision variant) in 1972 by Karp [Kar72] and was considered in cryptography as an interesting alternative to hardness assumptions based on number theory. Due to its simplicity, it was notably used in the 1980s, following [MH78], for the construction of several public-key encryption schemes.

Most of these proposals (if not all) were swiftly broken using lattice-based techniques (see [Odl90]), but the problem itself remains intractable for appropriate parameters and is even believed to be so for quantum computers. For instance, when the so-called density $d = n/\log_2(q)$ of the subset sum instance is close to 1 (i.e. $q \simeq 2^n$), the fastest known (classical and quantum) algorithms have complexity $2^{O(n)}$ (see [BBSS20] and references therein) and one can reach an alleged security level of λ bits with $n = \Theta(\lambda)$. Many cryptographic constructions were proposed whose security relies on the hardness of the subset sum problem: pseudo-random generators [IN96], bit commitments [IN96], public-key encryption [LPS10], ...

The concept of *zero-knowledge* proofs and arguments introduced in [GMR89] has become a fundamental tool in cryptography. It enables a prover to convince a verifier that some mathematical statement is true without revealing any additional information. Zero-knowledge proofs or arguments of knowledge, in which a prover demonstrates that they knows a "witness" of the validity of the statement, have found numerous applications in cryptography (notably for privacy-preserving constructions or to enforce honest behaviour of parties in complex protocols). The main goal of the present paper is to present new efficient zero-knowledge arguments of knowledge for the subset sum problem.

1.1 Prior Work

Given integers $w_1, \ldots, w_n$, t and q, an elegant zero-knowledge proof system due to Shamir [Sha86] (see also [BGKW90]) allows a prover to convince a verifier that they knows $x_1, \ldots, x_n \in \{0, 1\}$ such that the relation (1) holds. The proof system is combinatorial in nature and it requires $\Theta(\lambda)$ rounds of communication to achieve soundness error $2^{-\lambda}$ where each round requires $\Theta(n^2)$ bits of communication. For an alleged security level of λ bits, the overall communication complexity of Shamir's proof system is thus of $\Theta(\lambda^3)$. In [LNSW13], Ling, Nguyen, Stehlé, and Wang proposed a proof of knowledge of a solution for the infinity norm *inhomogeneous small integer solution* (ISIS) problem which is a vectorial variant of the subset sum problem. It is based on Stern's zero-knowledge proof of knowledge for the *syndrome decoding* problem [Ste94] and is also combinatorial. It thus requires a large number of rounds of communication and when specialized to the subset sum problem it also yields proofs with $\Theta(\lambda^3)$-bit communication complexity for an alleged security level of λ bits.

A secure multi-party computation (MPC) protocol allows a set of mutually distrusting parties to jointly evaluate a function f over their inputs while keeping those inputs private. An elegant approach to constructing zero-knowledge protocols has gained particular attention over the last years: the *MPC-in-the-head* paradigm of Ishai, Kushilevitz, Ostrovsky, and Sahai [IKOS09] in which a prover secretly shares their secret input, simulates the execution of an MPC protocol on these shares (in "their head"), commits to this execution and partially reveals it to the verifier on some challenge subset of parties. The verifier can then check that the partial execution is consistent and accepts or rejects accordingly. This approach was at first stood in the realm of theoretical cryptography (with a focus on the asymptotic performance for any problem in NP), but it was subsequently demonstrated to be also of practical relevance [GMO16,KKW18]. In [BD10], Bendlin and Damgård were the first to use the MPC-in-the-head paradigm in lattice-based cryptography. They proposed a zero-knowledge proof of knowledge of the plaintext contained in a given ciphertext from Regev's cryptosystem [Reg05] (and a variant they proposed). More recently, Baum and Nof [BN20] proposed an efficient zero-knowledge argument of knowledge of the *short integer solution* (SIS) problem (incorporating the *sacrificing* principle in the MPC-in-the-head paradigm). Beullens also recently proposed such arguments obtained from sigma protocols *with helper* [Beu20]. When applied to the subset sum problem itself, all (variants of) these protocols yield proofs with $\Theta(\lambda^3)$-bit communication complexity for an alleged security level of λ bits.

There exist numerous other protocols for (vectorial variants of) the subset sum problem from lattice-based cryptography. Until recently, they all introduce some slack in the proof, i.e. there is a difference between the language used for completeness and the language that the soundness guarantees (see, e.g. [BDLN16] for a generic argument of knowledge of a pre-image for *homomorphic one-way functions over integer vectors*). In particular, the witness that can be extracted from a proof is larger than the one that an honest prover uses (and in the subset sum problem, the extractor will not output a binary vector). This slack forces to use larger parameters for the underlying cryptosystem and induces some loss in efficiency. Conversely, we shall only consider exact arguments for the subset-sum problem in the present paper. Finally, new exact arguments were proposed recently [BLS19,ENS20,LNS21] but they require to use a modulus q of a special form (namely a prime number as in [BN20,Beu20] but with additional arithmetic constraints to make it "NTT-friendly").

1.2 Contributions

In the MPC-in-the-head paradigm, the prover wants to convince a verifier that they know a (secret) pre-image x of $y = f(x)$ for some one-way function f where the function f is represented as an arithmetic circuit. For the subset sum problem, the function f is defined *via* (1) and it is thus natural to consider the simple inner-product arithmetic circuit defined over $\mathbb{Z}_q$. The prover's secret input is the binary vector $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ and they have to perform some secret-sharing of x in $\mathbb{Z}_q$ in such a way that the shares of any unauthorized

set of parties should reveal no information about the secret. This approach has the major disadvantage that sharing a single bit requires several elements of $\mathbb{Z}_q$ each of size $\Theta(\lambda)$ bits.

We adapt this paradigm using a secret sharing scheme done directly over the integers. This approach was already used in cryptography (e.g. for multi-party computation modulo a shared secret modulus [CGH00]). To additively share a secret t in a given interval $[-T, T]$ for $T \in \mathbb{N}$, among $n \geq 2$ parties, a dealer may pick uniformly at random $t_1, \ldots, t_n \in [-T2^\rho, T2^\rho]$ under the constraint that $t = t_1 + \cdots + t_n$ (over the integers), for some parameter ρ. However, given $(n-1)$ shares, $t_2, \ldots, t_n$ for instance, the value $t_1 = t - (t_2 + \cdots t_n)$ is not randomly distributed in $[-T2^\rho, T2^\rho]$ and this may reveal information on the secret t. It is thus necessary to sample the shares in an interval sufficiently large in such a way that their distributions for distinct secrets are statistically indistinguishable. For a security level λ, this requires $\rho = \Omega(\lambda)$ and thus the additive sharing of bits involves shares of size $\Omega(\lambda)$. To overcome this limitation and use additive secret sharing over *small* integers, we will rely on *rejection*. The computation being actually simulated by the prover, they can abort the protocol whenever the sharing leaks information on the secret vector $x = (x_1, \ldots, x_n) \in \{0,1\}^n$. In some cases, the prover cannot respond to the challenge from the verifier and must abort the protocol. A similar idea was used for lattice-based signatures by Lyubashevsky [Lyu09]but using different methods.

Our technique also allows overcoming the second disadvantage of the previous tentatives to use the MPC-in-the-head paradigm for lattice-based problems. Indeed, using our additive secret sharing over the integers, we can prove the knowledge of some integer vector $x = (x_1, \ldots, x_n)$ satisfying relation (1) (for any q) and further prove that $x_i \in \{0,1\}$ for $i \in \{1, \ldots, n\}$. This is achieved by simulating a (single) non-linear operation modulo some arbitrary prime number q' (independent from q and much smaller than q). We also introduce another technique to prove that the solution $x = (x_1, \ldots, x_n)$ indeed lies in $\{0,1\}^n$ using some masking and a *cut-and-choose* strategy. Both methods yield zero-knowledge proofs with $\Theta(\lambda^2)$-bit communication complexity for an alleged security level of λ bits. This improvement is not only of theoretical interest since for $q \simeq 2^{256}$, our protocol can produce proof of size 13KB where Shamir's protocol [Sha86] (updated with modern tips) produces proof of size 1186KB and [LNSW13] produces proofs of size 2350KB.

Our protocols are particularly efficient for the subset sum problem where the modulus q is large. However, we show that our method has applications in other contexts in cryptography. We show that it can be used for the (binary) ISIS problem in lattice-based cryptography and that the resulting protocols are competitive with state-of-the-art protocols for this problem. We also present applications of our techniques to the context of *fully-homomorphic encryption* (FHE). Specifically, adaptations of our protocols provide efficient zero-knowledge arguments of plaintext and/or key knowledge for the so-called *Torus Fully Homomorphic Encryption* (TFHE) scheme from [CGGI20]. Eventually, we use our

technique to construct an efficient digital signature scheme based on a pseudo-random function due to Boneh, Halevi, and Howgrave-Graham [BHH01].

2 Preliminaries

2.1 Zero-Knowledge Proofs

A zero-knowledge (ZK) protocol for some polynomial-time decidable binary relation $\mathcal{R}$ (i.e., a relation that defines a language in NP) is defined by two probabilistic polynomial time (PPT) interactive algorithms, a prover $\mathcal{P}$ and a verifier $\mathcal{V}$: both $\mathcal{V}$ and $\mathcal{P}$ are given a common input x and $\mathcal{P}$ is given in addition a witness w such that $(x, w) \in \mathcal{R}$. Then, $\mathcal{P}$ and $\mathcal{V}$ exchange a sequence of messages alternatively until $\mathcal{V}$ outputs a bit b (with $b = 1$ indicating that $\mathcal{V}$ accepts $\mathcal{P}$'s claim and $b = 0$ indicating that $\mathcal{V}$ rejects the claim). The entire sequence of messages exchanged by $\mathcal{P}$ and $\mathcal{V}$, along with the answer b, is called a *transcript*.

A zero-knowledge argument for $\mathcal{R}$ with *soundness error* ϵ, *completeness error* α and (t, ζ)-*zero-knowledge* satisfies the following properties:

1. **Completeness:** if $(x, w) \in \mathcal{R}$, and $\mathcal{P}$ knows a witness w for x, they will succeed in convincing $\mathcal{V}$ (except with probability α), i.e.,
$$\Pr[\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle = 1] \geq 1 - \alpha.$$
2. **Soundness:** if there exists a PPT algorithm $\tilde{\mathcal{P}}$ such that
$$\tilde{\epsilon} := \Pr\left[\langle \tilde{\mathcal{P}}(x), \mathcal{V}(x) \rangle = 1\right] > \epsilon,$$
then there exists a PPT algorithm $\mathcal{E}$ (called the *extractor*) which, given rewindable black-box access to $\tilde{\mathcal{P}}$ outputs a witness w' for x in time in time $\mathsf{poly}(\lambda, (\tilde{\epsilon} - \epsilon)^{-1})$ with probability at least $1/2$.
3. **Zero-knowledge:** for every PPT algorithm $\tilde{\mathcal{V}}$, there exists a PPT algorithm $\mathcal{S}$ (called the *simulator*) which, given the input statement x and rewindable black-box access to $\tilde{\mathcal{V}}$, outputs a simulated transcript which is (t, ζ)-indistinguishable from $\mathrm{View}(\mathcal{P}(x, w), \tilde{\mathcal{V}}(x))$ (see the full version [FMRV22] for a formal definition).

Remark 1. The soundness property ensures that a PPT algorithm $\tilde{\mathcal{P}}$ without knowledge of the witness cannot convince $\mathcal{V}$ with probability greater than ϵ assuming that the underlying problem is hard. Otherwise, the existence of $\mathcal{E}$ implies that $\tilde{\mathcal{P}}$ can be used to compute a valid witness w' for x. If the zero-knowledge property holds only for the genuine verifier $\mathcal{V}$, then the protocol is deemed *honest-verifier* zero-knowledge.

2.2 MPC-in-the-Head and Batch Product Verification

The MPC-in-the-Head (MPCitH) paradigm [IKOS09] constructs ZK proofs from MPC protocols. Efficient instances of this paradigm have been published for the

first time these last years starting with a protocol called ZKBoo [GMO16] and has found numerous applications (e.g. [GMO16,KKW18,BN20]).

We consider a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ engaging a two-party interactive protocol for some public circuit C over a finite field $\mathbb{F}$ and some value $t \in \mathbb{F}$ such that $\mathcal{P}$ wants to convince $\mathcal{V}$ that they knows an $x \in \mathbb{F}$ satisfying $C(x) = t$.

In the MPCitH paradigm, the prover $\mathcal{P}$ usually decomposes their secret x into N shares $[\![x]\!]_1, ..., [\![x]\!]_N$ using some additive secret sharing over $\mathbb{F}$. Then, $\mathcal{P}$ simulates an N-party MPC protocol for evaluating C. At the end of the MPC protocol, using a commitment scheme (see definition in the full version [FMRV22]), $\mathcal{P}$ commits to the N views of the parties resulting from the MPC protocol simulation. $\mathcal{V}$ then challenges $\mathcal{P}$ to open a subset of the views. $\mathcal{P}$ answers by opening these views and $\mathcal{V}$ checks that these views are consistent with the MPC process as well as valid openings of the commitments. In the basic setting where $N - 1$ out of N parties are opened, the resulting zero-knowledge protocol achieves a soundness error of $1/N$.

Batch Product Verification. Using the MPCitH approach the linear operations over $\mathbb{F}$ (i.e. addition in $\mathbb{F}$ and multiplication by constants in $\mathbb{F}$) can be handled easily and are almost free in terms of computation and communication. The most cumbersome part of the MPCitH method is to handle non-linear operations and in particular multiplications in $\mathbb{F}$. The authors of [BN20] propose an MPC protocol to verify the correctness of a product in $\mathbb{F}$ by "sacrificing" another one. This construction enables to check that a triple of sharings $([\![x]\!], [\![y]\!], [\![z]\!])$ is such that $x \cdot y = z$, by using a second random triple $([\![a]\!], [\![b]\!], [\![c]\!])$ satisfying $a \cdot b = c$. The second triple can be used a single time (to preserve the zero-knowledge property), hence the "sacrifice".

Recently [KZ22] has adapted and optimized this method to build an efficient MPC protocol which check simultaneously many products by sacrificing a dot-product (see the full version [FMRV22]).

Additive Sharing. In most recent MPCitH schemes, in order to decrease the communication costs, when the prover splits their secret x into N shares $[\![x]\!]_1, ..., [\![x]\!]_N$, the first $N - 1$ shares are generated using a pseudo-random generator and only the N-th share $[\![x]\!]_N$ is computed in such a way that $x = [\![x]\!]_1 + \cdots + [\![x]\!]_N$ in $\mathbb{F}$. In this paper, since our sharings will not be defined over some additive group, we will generate the N shares $[\![x]\!]_1, ..., [\![x]\!]_N$ from N seeds using a pseudo-random generator and we will introduce an auxiliary value Δx (not distributed over the same set) such that $x = [\![x]\!]_1 + \cdots + [\![x]\!]_N + \Delta x$ over the integers.

3 General Idea

We consider an instance $(w, t) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ of the subset sum problem (SSP) and denote x one solution. We have $x \in \{0, 1\}^n$ and $\sum_{j=1}^{n} x_j \cdot w_j = t \bmod q$.

We want to use the MPCitH paradigm to build a zero-knowledge protocol that proves the knowledge of a solution for the instance (w, t). To proceed, we need to build an MPC protocol with honest-but-curious parties taking as inputs shares of the secret x, and possibly shares of other data, and which computation can only succeed if x is a valid solution of the SSP instance. As a first ingredient, we need a method to share the secret x between the different parties.

3.1 The Naive Approach

The SSP instance is defined on $\mathbb{Z}_q$, so a natural sharing of x would be defined as:

$$\begin{cases} [\![x]\!]_i \xleftarrow{\$} (\mathbb{Z}_q)^n \text{ for all } i \in [N], \\ \Delta x \leftarrow x - \sum_{i=1}^{N} [\![x]\!]_i \bmod q \end{cases}.$$

In the MPCitH paradigm, the communication cost of a sharing is the cost to send the auxiliary values, *i.e.* the vector Δx. Here, the natural sharing of x costs

$$n \cdot \log_2(q) \text{ bits.}$$

If we take $n = 256$ and $q = 2^{256}$, the cost is about 2^{16} bits = 8 KB. To achieve a soundness error of 2^{-128} with $N = 256$, we need to repeat the protocol at least 16 times, so the communication cost of the protocol would be already more than 128 KB for the sole sharing of x (some communication being further required for the MPCitH protocol). Asymptotically, the parameters for the subset sum problem are chosen such that $n = \Theta(\lambda)$ and $\log_2 q = \Theta(\lambda)$, the communication cost of this sharing is thus about $\Theta(\lambda^2)$ bytes per protocol repetition. Since we need to repeat the protocol about $\Theta(\lambda)$ times to achieve a $2^{-\lambda}$ soundness error the global communication cost is then of at least $\Theta(\lambda^3)$ (for the sharing only).

We present hereafter an alternative strategy for the sharing of x, which achieves better practical and asymptotic communication costs.

3.2 Sharing on the Integers and Opening with Abort

We propose another way to share the secret x to achieve lower communication. We know that x is a binary vector (*i.e.* $x \in \{0, 1\}^n$), so instead of the natural sharing, we suggest to use a sharing defined on the integers, that is

$$\begin{cases} [\![x]\!]_i \xleftarrow{\$} \{0, \ldots, A-1\}^n \text{ for all } i \in [N], \\ \Delta x \leftarrow x - \sum_{i=1}^{N} [\![x]\!]_i. \end{cases}$$

However, this sharing leaks information about the secret x. The distribution Δx_j is not the same depending on whether $x_j = 0$ or $x_j = 1$ as illustrated on Figure 1. To solve this issue, the prover must abort the protocol in some cases.

To see how this leakage can be effectively exploited to (partly) recover x, let us recall that at the end of the protocol, the verifier shall ask the prover to open

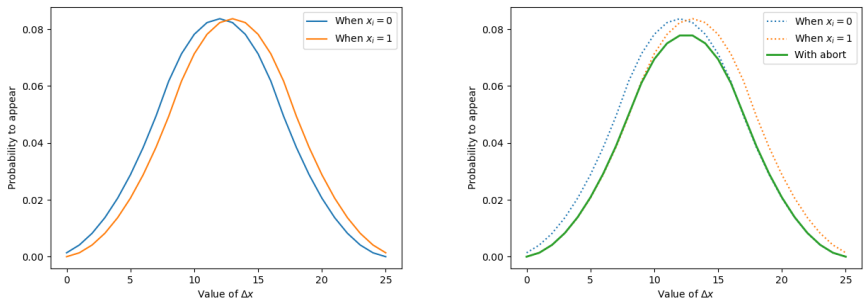

Fig. 1: Probability mass function of Δx_j when $x_j = 0$ and when $x_j = 1$ (on the left) and of Δx_j with abort (on the right), for $N = 3$ and $A = 9$.

the views of all parties except one. Let us denote i^* the index of the unopened party. It means the verifier will have access to

$$\{[\![x]\!]_i\}_{i \neq i^*} \quad \text{and} \quad \Delta x \ .$$

For the sake of simplicity, let us first consider the case $n = 1$, *i.e.* $x \in \{0, 1\}$ and $[\![x]\!]$ is the sharing of a single integer. With the opened values, the verifier can compute

$$x - [\![x]\!]_{i^*} \quad \text{as} \quad \Delta x + \sum_{i \neq i^*} [\![x]\!]_i \ .$$

Now let us denote $Y = x - [\![x]\!]_{i^*}$ the underlying random variable over the uniform random sampling of $[\![x]\!]_{i^*}$. We have

$$\Pr(Y = -A + 1) = \begin{cases} \frac{1}{A} & \text{if } x = 0 \\ 0 & \text{if } x = 1 \end{cases} \quad \text{and} \quad \Pr(Y = 1) = \begin{cases} 0 & \text{if } x = 0 \\ \frac{1}{A} & \text{if } x = 1 \end{cases}$$

while

$$\Pr(Y = y) = \frac{1}{A} \quad \text{for every } y \in \{-A + 2, \ldots, 0\} \ .$$

So by observing $x - [\![x]\!]_{i^*} = -A + 1$ one learns $(x, [\![x]\!]_{i^*}) = (0, -A + 1)$. Similarly, by observing $x - [\![x]\!]_{i^*} = 1$ one learns $(x, [\![x]\!]_{i^*}) = (1, 0)$. To avoid this flaw, the prover must abort the protocol before revealing $\{[\![x]\!]_i\}_{i \neq i^*}$ and Δx whenever one of these two cases occurs. This notably implies that Δx must not be revealed before receiving the challenge i^*, but it should still be committed beforehand in order to ensure the soundness of the protocol. Doing so, we modify the distribution of the revealed auxiliary value which does not leak any information about x anymore as illustrated in Figure 1, and the probability to abort does not leak information about x since it is $1/A$ in the both cases ($x = 0$ and $x = 1$).

Let us now come back to the general case of $n \geq 1$. The prover applies the above abortion strategy for all the coordinates of x, namely

- if there exists $j \in [n]$ such that $x_j = 0$ and $[\![x_j]\!]_{i^*} = A-1$, the prover aborts;
- if there exists $j \in [n]$ such that $x_j = 1$ and $[\![x_j]\!]_{i^*} = 0$, the prover aborts;
- otherwise the prover proceeds.

The probability to abort, which we call *rejection rate*, is

$$1 - \left(1 - \frac{1}{A}\right)^n \leq \frac{n}{A} \ .$$

We note that the rejection rate can be tightly approximated by the n/A upper bound when A is sufficiently large. In order to achieve a small (constant) rejection rate, we should hence choose A greater than n. Asymptotically, we then have $A = \Theta(n) = \Theta(\lambda)$, which represents an exponential improvement compared to $q = 2^{\Theta(\lambda)}$.

Let us now analyze the computation cost of our strategy for sharing x. In the absence of rejection, Δx_j belongs to $\{-N \cdot (A-1)+1, \ldots, 0\}$, therefore sending the auxiliary value Δx would cost $n \cdot \log_2(N \cdot (A-1))$ bits. However, the prover can save communication by sending $x - [\![x]\!]_{i^*}$ instead, which is strictly equivalent in terms of revealed information by the relation $x - [\![x]\!]_{i^*} = \Delta x + \sum_{i \neq i^*} [\![x]\!]_i$. Since each coordinate of $x - [\![x]\!]_{i^*}$ is uniformly distributed over $\{-A+2, \ldots, 0\}$, sending it only costs

$$n \cdot \log_2(A-1) \text{ bits.}$$

With $x - [\![x]\!]_{i^*}$, the verifier can recover Δx by computing $\Delta x = (x - [\![x]\!]_{i^*}) - \sum_{i \neq i^*} [\![x]\!]_i$. The cost of this sharing has the advantage of being independent of the modulus q on which the SSP instance is defined. The value of A will be chosen according to the desired trade-off between communication cost and rejection rate. If $n = 256$ and $A = 2^{16}$, we have a cost of 0.5 KB for a rejection rate of 0.0038, which is much better than the 8 KB of the naive approach.

Let us remark that adding an abort event does not impact the soundness of the protocol. A malicious prover can abort as many times she wants claiming that it would leak information, but an abortion does not help to convince the verifier. The soundness theorem will state that someone who does not know the secret can only answer with a probability smaller than the constant value called soundness error, and adding an abort event cannot increase this probability. The prover could sample a random party i' and give to i' a wrong share and she may indeed decide to abort if the verifier challenge is not i', but this does not change the fact that the probability for the prover to convince the verifier is the probability that the prover guesses the verifier challenge a priori.

Now that we have defined the sharing of x, we need to demonstrate two properties of the shared SSP instance through multi-party computation. The first one is the SSP relation which in the shared setting translates to

$$\sum_{j=1}^{n} [\![x_j]\!] \cdot w_j = [\![t]\!] \bmod q$$

for a sharing $[\![t]\!]$ of t. The linearity of this relation makes it easy to deal with: the share $[\![t]\!]_i$ can simply be computed as $[\![t]\!]_i := \sum_{j=1}^{n} [\![x_j]\!]_i \cdot w_j \bmod q$ and

committed to the verifier by each party. The verifier can then check that the open parties have correctly computed their shares $[\![t]\!]_i$ and that the relation $\sum_{i=1}^{N} [\![t]\!]_i = [\![t]\!] \bmod q$ well holds. The second property which must be demonstrated through multi-party computation is that the solution x corresponding to the sharing $[\![x]\!]$ is a binary vector. This is not a priori guaranteed to the verifier since the shares of the coordinate of x are defined over $\{0, \ldots, A-1\}$ and the correctness of the linear relation does not imply that x is indeed binary. We present two different solutions to this issue in the following.

3.3 Binarity Proof from Batch Product Verification

Our first solution relies on standard MPC-in-the-Head techniques to prove the relation

$$x \circ (x-1) = 0$$

where $\circ$ denotes the coordinate-wise product, 0 and 1 are to be interpreted as the all-0 and all-1 vectors. To this aim, we can use the MPC-in-the-Head batch product verification suggested in [LN17,BN20] and recently improved in [KZ22] (see Section 2.2). However, we can do better than a straight application of those techniques.

The relation $x \circ (x-1) = 0$ is defined in $\mathbb{Z}_q$ and the above techniques imply to send at least one field element per product, that is n elements from $\mathbb{Z}_q$. To save communication and since the sharing $[\![x]\!]$ is defined on the integers, we can work on a smaller field. We previously explained that the verifier receives $\{[\![x]\!]_i\}_{i \neq i^*}$ and Δx from the prover, so they can check that, for all $j \in [n]$,

$$-A + 2 \leq x_j - [\![x_j]\!]_{i^*} \leq 0 \ .$$

They further trusts $[\![x_j]\!]_{i^*} \in \{0, \ldots, A-1\}$ (which is verified for the open parties). Thus the verifier can deduce that, for all $j \in [n]$,

$$-A + 2 \leq x_j \leq A - 1 \ . \tag{2}$$

Let q' be a prime such that $q' \geq A$. If the prover convinces the verifier that $x_j(x_j - 1) = 0 \bmod q'$, then the latter deduces that $x_j \in \{0, 1\}$ because

$$\begin{aligned} q' | x_j(x_j-1) \ &\Rightarrow \ (q' | x_j) \text{ or } (q' | x_j - 1) \\ &\Rightarrow \ (x_j = 0) \text{ or } (x_j = 1) \quad \text{by (2)} \end{aligned}$$

The prover hence just needs to prove $x \circ (x-1) = 0 \bmod q'$ for some prime q' such that $q' \geq A$. To this purpose, we apply the batch product verification of [KZ22] as follows (see also the full version [FMRV22]).

The prover first samples $a \in (\mathbb{Z}_{q'})^n$ with its sharing

$$[\![a]\!]_i \xleftarrow{\$} (\mathbb{Z}_{q'})^n \text{ for } i \in [N] \ .$$

The value a is hence defined as a uniform random element of $(\mathbb{Z}_{q'})^n$ and no auxiliary value Δa is necessary. The prover then computes $c = \langle a, x \rangle$ and its sharing as

$$\begin{cases} [\![c]\!]_i \xleftarrow{\$} \mathbb{Z}_{q'} \text{ for all } i \in [N], \\ \Delta c \xleftarrow{\$} c - \sum_{i=1}^{N} [\![c]\!]_i \bmod q' \end{cases} .$$

The prover gives the shares of x, a and c as inputs to the parties and runs the following MPC protocol:

1. the parties get a random challenge $\varepsilon \in (\mathbb{Z}_{q'})^n$ from the verifier;
2. the parties locally set $[\![\alpha]\!] = \varepsilon \circ (1 - [\![x]\!]) + [\![a]\!]$;
3. the parties open $[\![\alpha]\!]$ to get α;
4. the parties locally set $[\![v]\!] = \langle \alpha, [\![x]\!] \rangle - [\![c]\!]$;
5. the parties open $[\![v]\!]$ to get v;
6. the parties accept iff $v = 0$.

Besides the input shares and commitments, the prover-to-verifier communication cost of the corresponding MPCitH zero-knowledge protocol only results from the size of $[\![\alpha]\!]_{i^*}$ (the broadcasted vector of the unopened party i^*), which is of

$$n \cdot \log_2(q') \text{ bits.}$$

We stress that the prover does not need to send $[\![v]\!]_{i^*}$ because the verifier knows that v must be zero and will deduce $[\![v]\!]_{i^*} = -\Delta v - \sum_{i \neq i^*} [\![v]\!]_i$

As described in Section 2.2, the batch product MPC verification produces false positives with probability $1/q'$. Thus the soundness error of the obtained zero-knowledge protocol is

$$1 - \Big(1 - \frac{1}{N}\Big)\Big(1 - \frac{1}{q'}\Big) < \frac{1}{N} + \frac{1}{q'} .$$

On the other hand, the protocol has a rejection rate of $1 - (1 - \frac{1}{A})^n$ and a prover-to-verifier communication cost (in bits) of

$$2 \cdot (2\lambda) + \underbrace{n \cdot \log_2(A-1)}_{x - [\![x]\!]_{i^*}} + \underbrace{n \cdot \log_2(q')}_{\Delta\alpha} + \underbrace{\log_2(q')}_{\Delta c} + \lambda \log_2 N + 2\lambda .$$

3.4 Binarity Proof from Masking and Cut-and-Choose Strategy

Our second solution to prove that $[\![x]\!]$ encodes a binary vector relies on a masking of x and a cut-and-choose strategy. The idea is to generate a random vector r from $\{0,1\}^n$ and to apply the sharing described in Section 3.2 to r. In addition, the prover computes (and commits) $\tilde{x} := x \oplus r \in \{0,1\}^n$ where $\oplus$ represents the XOR operation. Instead of giving the shares $[\![x]\!]$ of x as inputs of the MPC protocol, the idea is now to send the shares $[\![r]\!]$ of r. Then using $\tilde{x}$, the parties can locally deduce a sharing of x as

$$[\![x]\!] = (1 - \tilde{x}) \circ [\![r]\!] + \tilde{x} \circ (1 - [\![r]\!])$$

which is a linear relation in $[\![r]\!]$, and the verifier can further deduce the auxiliary value Δx from Δr as

$$\Delta x = (1 - \tilde{x}) \circ \Delta r + \tilde{x} \circ (1 - \Delta r) \ .$$

By replacing $[\![x]\!]$ with $[\![r]\!]$ the parties' input is made independent of the secret. The interest of doing so is to enable a cut-and-choose strategy to prove that $[\![r]\!]$ encodes a binary vector, which in turns implies that $x = \tilde{x} \oplus r$ is a binary vector. More precisely, at the beginning of the zero-knowledge protocol, the prover produces M binary vectors $r^{[\ell]}$ and their corresponding shares $[\![r^{[\ell]}]\!]$ (in practice these vectors and their sharings are pseudo-randomly derived from some seeds). Then the prover commits those sharings $[\![r^{[\ell]}]\!]$ as well as the corresponding masked vectors $\tilde{x}^{[\ell]} := x \oplus r^{[\ell]}$. Then the verifier asks to open all the sharings $r^{[\ell]}$ except one and checks that they correspond to binary vectors. The verifier will hence trust that the unopened sharing encodes also a binary vector with a soundness error of $1/M$. We stress that all the values $\tilde{x}^{[\ell]}$ for which $r^{[\ell]}$ is opened must remain hidden (otherwise x could be readily recovered). The obtained zero-knowledge protocol has a soundness error of

$$\max\left\{\frac{1}{M}, \frac{1}{N}\right\} \ ,$$

a rejection rate of $1 - (1 - \frac{1}{A})^n$ and a prover-to-verifier communication cost (in bits) of

$$2 \cdot (2\lambda) + \underbrace{\lambda \log_2 M}_{\text{Cost of C\&C}} + \underbrace{n \cdot \log_2(A-1)}_{r - [\![r]\!]_{i^*}} + \underbrace{n}_{\tilde{x}} + \lambda \log_2 N + 2\lambda \ .$$

3.5 Asymptotic Analysis

We analyze hereafter the asymptotic complexity of the two variants of our protocol. We show that for a security parameter λ both variants have an asymptotic communication cost of $\Theta(\lambda^2)$ and an asymptotic computation time of $\Theta(\lambda^4)$.

For the binarity proof based on masking and cut-and-choose, we assume $M = N$ (which is optimal for the communication cost given the soundness error). For the other parameters, let us recall that

- for a security parameter λ, one must take $n \approx \log_2 q = \Theta(\lambda)$,
- the prime q' can be chosen as the smallest prime greater than A, which implies $q' \approx A$.

For both variants, the asymptotic communication cost for one repetition of the protocol is then of

$$\Theta(\lambda \log_2 A + \lambda \log_2 N) \ .$$

Since each repetition has a soundness error of $\Theta(1/N)$, the protocol must be repeated $\tau = \Theta(\lambda / \log_2 N)$ times to reach a global soundness error of $2^{-\lambda}$. The probability that any of these τ repetitions aborts is given by

$$1 - \left(1 - \frac{1}{A}\right)^{n \cdot \tau} \approx \frac{n \cdot \tau}{A}$$

where the approximation is tight when A is sufficiently large. Thus for a small constant rejection probability, one must take $A = \Theta(n \cdot \tau) = \Theta(\lambda^2 / \log_2 N)$. We have a communication cost for the τ iterations in

$$\Theta\left(\lambda^2 \frac{\log_2 A}{\log_2 N} + \lambda^2\right) = \Theta\left(\frac{\lambda^2}{\log_2 N} \log_2\left(\frac{\lambda^2}{\log_2 N}\right) + \lambda^2\right)$$

and we hence obtain a minimal asymptotic communication cost of $\Theta(\lambda^2)$ by taking $N = \Theta(\lambda)$.

The asymptotic computation time for one repetition of the protocol is of $\Theta(Nn(\log_2 q)(\log_2 A))$, where the term $(\log_2 q)(\log_2 A)$ arises from the complexity of the multiplication between an element of $\mathbb{Z}_q$ and a value smaller than A. We hence get a computation time of $\Theta(\lambda^3 \log_2 \lambda)$ per repetition which makes $\Theta(\lambda^4)$ for τ repetitions.

4 Protocols and Security Proofs

In this section, we formally describe our two protocols and state their security. We further introduce a method to decrease the rejection rate.

4.1 Protocol with Batch Product Verification

Protocol description. In Section 3.3, we proposed an MPC protocol that proves that the sharing $[\![x]\!]$ encodes a binary vector. We then add the checking of the linear relation as described in Section 3.2 and we transform the multi-party computation into a zero-knowledge protocol which proves the knowledge of a solution of an SSP instance. We give the formal description of our protocol in Protocol 1. The protocol makes use of a pseudo-random generator PRG, a tree-based pseudo-random generator TreePRG (see definition in [KKW18]), two collision-resistant hash functions Hash_i for $i \in \{1, 2\}$ and a commitment scheme $(\mathrm{Com}, \mathrm{Verif})$ as defined in the full version [FMRV22]. In this description, the procedure Check returns 0 if the evaluated condition is false (*i.e.* the equality does not hold) and the execution continues otherwise.

Security proofs. The following theorems state the completeness, zero-knowledge and soundness of Protocol 1. The proofs of Theorems 1, 2 and 3 are provided in appendix of the full version [FMRV22].

Theorem 1 (Completeness). *A prover $\mathcal{P}$ who knows a solution x to the subset sum instance $(w, t) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ and who follows the steps of Protocol 1 convinces the verifier $\mathcal{V}$ with probability*

$$\left(1 - \frac{1}{A}\right)^n .$$

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
$x \in \{0,1\}^n$		
$w \in \mathbb{Z}_q^n, t = \langle w, x \rangle$		w, t
$\mathsf{mseed} \xleftarrow{\$} \{0,1\}^\lambda$		
Compute parties' seeds		
$(\mathsf{seed}_1, \rho_1), \ldots, (\mathsf{seed}_N, \rho_N)$		
with TreePRG(mseed)		
For each party $i \in \{1, \ldots, N\}$:		
$[\![a]\!]_i, [\![x]\!]_i, [\![c]\!]_i \leftarrow \mathrm{PRG}(\mathsf{seed}_i)$		$\triangleright\ a \in \mathbb{Z}_{q'}^n,\ c \in \mathbb{Z}_{q'},\ [\![x]\!]_i \in \{0, \ldots, A-1\}^n$
$\mathsf{com}_i = \mathrm{Com}(\mathsf{seed}_i; \rho_i)$		
$\Delta x = x - \sum_i [\![x]\!]_i$		
$\Delta c = \langle a, x \rangle - \sum_i [\![c]\!]_i$		
$h = \mathrm{Hash}_1(\Delta x, \Delta c, \mathsf{com}_1, \ldots, \mathsf{com}_N)$		
	$\xrightarrow{h}$	
		$\varepsilon \xleftarrow{\$} \mathbb{Z}_{q'}^n$
	$\xleftarrow{\varepsilon}$	
The parties locally set		
- $[\![t]\!] = \langle w, [\![x]\!] \rangle$		$\triangleright\ t \in \mathbb{Z}_q$
- $[\![\alpha]\!] = \varepsilon \circ (1 - [\![x]\!]) + [\![a]\!]$		$\triangleright\ \alpha \in \mathbb{Z}_{q'}^n$ (computation in $\mathbb{Z}_{q'}$)
The parties open $[\![\alpha]\!]$ to get α.		
The parties locally set		
$[\![v]\!] = \langle \alpha, [\![x]\!] \rangle - [\![c]\!]$		$\triangleright\ v \in \mathbb{Z}_{q'}$ (computation in $\mathbb{Z}_{q'}$)
$h' = \mathrm{Hash}_2([\![t]\!], [\![\alpha]\!], [\![v]\!])$		
	$\xrightarrow{h'}$	
		$i^* \xleftarrow{\$} \{1, \ldots, N\}$
	$\xleftarrow{i^*}$	
If there exists $j \in [n]$ such that:		
- either $[\![x_j]\!]_{i^*} = 0$ with $x_j = 1$		
- or $[\![x_j]\!]_{i^*} = A - 1$ with $x_j = 0$,		
then abort.		
$y = x - [\![x]\!]_{i^*}$		
	$\xrightarrow{(\mathsf{seed}_i, \rho_i)_{i \neq i^*},\ \mathsf{com}_{i^*},\ y,\ \Delta c,\ [\![\alpha]\!]_{i^*}}$	
		For all $i \neq i^*$,
		$[\![a]\!]_i, [\![x]\!]_i, [\![c]\!]_i \leftarrow \mathrm{PRG}(\mathsf{seed}_i)$
		$\Delta x = y - \sum_{i \neq i^*} [\![x]\!]_i$
		$\Delta \alpha = \varepsilon \cdot (1 - \Delta x)$
		For all $i \neq i^*$,
		Rerun the party i as the prover
		and compute the commitment com_i.
		$\Delta t = \langle w, \Delta x \rangle$
		$\Delta v = \langle \alpha, \Delta x \rangle - \Delta c$
		$[\![t]\!]_{i^*} = t - \Delta t - \sum_{i \neq i^*} [\![t]\!]_i$
		$[\![v]\!]_{i^*} = -\Delta v - \sum_{i \neq i^*} [\![v]\!]_i$
		Check $h = \mathrm{Hash}_1(\Delta x, \Delta c, \mathsf{com}_1, \ldots, \mathsf{com}_N)$
		Check $h' = \mathrm{Hash}_2([\![t]\!], [\![\alpha]\!], [\![v]\!])$
		Return 1

Protocol 1: Zero-knowledge argument for Subset Sum Problem via MPC-in-the-head with rejection, using batch product verification to prove binarity.

Theorem 2 (Zero-Knowledge). *Let the PRG used in Protocol 1 be (t, ε_{PRG})-secure and the commitment scheme* Com *be (t, ε_{Com})-hiding. There exists an effi-*

cient simulator $\mathcal{S}$ which outputs a transcript which is $(t, \varepsilon_{PRG}+\varepsilon_{Com})$-indistinguishable from a real transcript of Protocol 1.

Theorem 3 (Soundness). *Suppose that there is an efficient prover $\tilde{\mathcal{P}}$ that, on input (w,t), convinces the honest verifier $\mathcal{V}$ on input H, y to accept with probability*

$$\tilde{\epsilon} := \Pr[\langle \tilde{\mathcal{P}}(w,t), \mathcal{V}(w,t)\rangle = 1] > \epsilon$$

for a soundness error ϵ equal to

$$\frac{1}{q'} + \frac{1}{N} - \frac{1}{q'} \cdot \frac{1}{N} \ .$$

Then, there exists an efficient probabilistic extraction algorithm $\mathcal{E}$ that, given rewindable black-box access to $\tilde{\mathcal{P}}$, produces either a witness x such that $t = \langle w, x\rangle$ and $x \in \{0,1\}^n$, or a commitment collision, by making an average number of calls to $\tilde{\mathcal{P}}$ which is upper bounded by

$$\frac{4}{\tilde{\epsilon} - \epsilon} \cdot \left(1 + \tilde{\epsilon} \cdot \frac{2 \cdot \ln(2)}{\tilde{\epsilon} - \epsilon}\right) \ .$$

Proof size. To achieve a targeted soundness error $2^{-\lambda}$, we can perform τ parallel executions of the protocol such that $\epsilon^\tau \leq 2^{-\lambda}$. Such parallel repetition does not preserve (general) zero-knowledge and the resulting scheme achieves *honest verifier* zero knowledge. And instead of sending τ values for h and h', the prover can merge them together to send a single h and a single h'. Moreover, instead to sending the $N-1$ seeds and commitment randomness of $(\mathsf{seed}_i, \rho_i)_{i \neq i^*}$ for each execution, we can instead send the sibling path from $(\mathsf{seed}_{i^*}, \rho_{i^*})$ to the tree root, it costs at most $\lambda \cdot \log_2(N)$ bits (we need to reveal $\log_2(N)$ nodes of the tree) by execution. The communication cost (in bits) of the protocol with τ repetitions is

$$\textsc{Size} \ = \ 4\lambda \ + \ \tau \ \cdot \ [n \cdot (\log_2(A-1) + \log_2(q')) + \log_2(q') + \lambda \log_2 N + 2\lambda]$$

while the soundness error and rejection rate scale as

$$\left(\frac{1}{q'} + \frac{1}{N} - \frac{1}{q'} \cdot \frac{1}{N}\right)^\tau \quad \text{and} \quad 1 - \left(1 - \frac{1}{A}\right)^{\tau \cdot n}$$

respectively. Let us stress that the obtained size is independent of the modulus q (and of the size of the integers $\{w_j\}, t$).

4.2 Protocol with Cut-and-Choose Strategy

Protocol description. As described in Section 3.4, we can also use a cut-and-choose strategy to prove that the vector $[\![x]\!]$ is binary. It is possible since we can remplace the input $[\![x]\!]$ of the multi-party computation by a sharing $[\![r]\!]$ *independent* of the secret, where r is a mask uniformly sampled in $\{0,1\}^n$. To

achieve a targeted soundness error $2^{-\lambda}$, we can perform τ parallel executions of the protocol such that $\epsilon^\tau \leq 2^{-\lambda}$. Like [KKW18], instead of performing τ independent cut-and-choose phases each resulting in trusting one sharing $[\![r]\!]$ among M, we can perform a global cut-and-choose phase resulting in τ trusted sharings $[\![r]\!]$ among a larger M (see [KKW18] for more details). We give the formal description of this zero-knowledge protocol in Protocol 2. The protocol makes use of a pseudo-random generator PRG, a tree-based pseudo-random generator TreePRG (see definition in [KKW18]), four collision-resistant hash functions Hash_i for $i \in \{1, 2, 3, 4\}$ and a commitment scheme $(\mathrm{Com}, \mathrm{Verif})$ as defined in the full version [FMRV22]. In this description, the procedure Check returns 0 if the evaluated condition is false (*i.e.* the equality does not hold) and the execution continues otherwise.

Security proofs. The following theorems state the completeness, zero-knowledge and soundness of Protocol 2. The proofs of Theorems 4, 5 and 6 are provided in appendix of the full version [FMRV22].

Theorem 4 (Completeness). *A prover $\mathcal{P}$ who knows a solution x to the subset sum instance $(w, t) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ and who follows the steps of Protocol 2 convinces the verifier $\mathcal{V}$ with probability*

$$\left(1 - \frac{1}{A}\right)^{\tau \cdot n} .$$

Theorem 5 (Honest-Verifier Zero-Knowledge). *Let the PRG used in Protocol 2 be (t, ε_{PRG})-secure and the commitment scheme* Com *be (t, ε_{Com})-hiding. There exists an efficient simulator $\mathcal{S}$ which, given random challenges J and L outputs a transcript which is $(t, \tau \cdot \varepsilon_{PRG} + \tau \cdot \varepsilon_{Com})$-indistinguishable from a real transcript of Protocol 2.*

Theorem 6 (Soundness). *Suppose that there is an efficient prover $\tilde{\mathcal{P}}$ that, on input (w, t), convinces the honest verifier $\mathcal{V}$ on input H, y to accept with probability*

$$\tilde{\epsilon} := \Pr[\langle \tilde{\mathcal{P}}(w, t), \mathcal{V}(w, t) \rangle = 1] > \epsilon$$

for a soundness error ϵ equal to

$$\max_{M-\tau \leq k \leq M} \left\{ \frac{\binom{k}{M-\tau}}{\binom{M}{M-\tau} \cdot N^{k-M+\tau}} \right\} .$$

Then, there exists an efficient probabilistic extraction algorithm $\mathcal{E}$ that, given rewindable black-box access to $\tilde{\mathcal{P}}$, produces either a witness x such that $t = \langle w, x \rangle$ and $x \in \{0, 1\}^n$, or a commitment collision, by making an average number of calls to $\tilde{\mathcal{P}}$ which is upper bounded by

$$\frac{4}{\tilde{\epsilon} - \epsilon} \cdot \left(1 + \tilde{\epsilon} \cdot \frac{8 \cdot M}{\tilde{\epsilon} - \epsilon}\right) .$$

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
$x \in \{0,1\}^n$		
$w \in \mathbb{Z}_q^n, t = \langle w, x \rangle$		w, t
$\mathsf{mseed}^{[0]} \xleftarrow{\$} \{0,1\}^\lambda$		
$(\mathsf{mseed}^{[e]})_{e\in[M]} \leftarrow \text{TreePRG}(\mathsf{mseed}^{[0]})$		
For each $e \in \{1,\dots,M\}$:		
$r^{[e]} \leftarrow \text{PRG}(\mathsf{mseed}^{[e]})$		$\triangleright\ r^{[e]} \in \{0,1\}^n$
$(\mathsf{seed}_i^{[e]}, \rho_i^{[e]})_{i\in[N]} \leftarrow \text{TreePRG}(\mathsf{mseed}^{[e]})$		
For each $i \in \{1,\dots,N\}$:		
$[\![r^{[e]}]\!]_i \leftarrow \text{PRG}(\mathsf{seed}_i^{[e]})$		$\triangleright\ [\![r^{[e]}]\!]_i \in \{0,\dots,A-1\}^n$
$\mathsf{com}_i^{[e]} = \text{Com}(\mathsf{seed}_i^{[e]}; \rho_i^{[e]})$		
$\Delta r^{[e]} = r^{[e]} - \sum_i [\![r^{[e]}]\!]_i$		
$h_e = \text{Hash}_1(\Delta r^{[e]}, \mathsf{com}_1^{[e]}, \dots, \mathsf{com}_n^{[e]})$		
$h = \text{Hash}_2(h_1, \dots, h_M)$		
	$\xrightarrow{h}$	
		$J \xleftarrow{\$} \{J \subset [M]\ ;\ \lvert J\rvert = \tau\}$
	$\xleftarrow{J}$	
For each $e \in J$:		
$\tilde{x}^{[e]} = x \oplus r^{[e]}$		$\triangleright\ \oplus$ is the XOR operation ($\tilde{x} \in \{0,1\}^n$)
The parties locally set		
$[\![x^{[e]}]\!] = (1 - \tilde{x}^{[e]}) \circ [\![r^{[e]}]\!]$		
$+\tilde{x}^{[e]} \circ (1 - [\![r^{[e]}]\!])$		
and they set $[\![t^{[e]}]\!] = \langle w, [\![x^{[e]}]\!] \rangle$.		
$h'_e = \text{Hash}_3(\tilde{x}^{[e]}, [\![t^{[e]}]\!])$		
$h' = \text{Hash}_4((h'_e)_{e\in J})$		
	$\xrightarrow{h',\ (\mathsf{mseed}^{[e]})_{e\in[M]\setminus J}}$	
		$L = \{\ell_e\}_{e\in J} \xleftarrow{\$} \{1,\dots,N\}^\tau$
	$\xleftarrow{L}$	
If there exists $(e,j) \in J \times [n]$ such that:		
- either $[\![r_j^{[e]}]\!]_{\ell_e} = 0$ with $r_j^{[e]} = 1$		
- or $[\![r_j^{[e]}]\!]_{\ell_e} = A-1$ with $r_j^{[e]} = 0$,		
then abort.		
$y = r^{[e]} - [\![r^{[e]}]\!]_{\ell_e}$		
	$\xrightarrow{\left(\begin{array}{c}(\mathsf{seed}_i^{[e]}, \rho_i^{[e]})_{i\neq\ell_e} \\ y, \tilde{x}^{[e]},\ \mathsf{com}_{\ell_e}^{[e]}\end{array}\right)_{e\in J}}$	
		For each $e \notin J$:
		Compute h_e using $\mathsf{mseed}^{[e]}$
		For each $e \in J$:
		For all $i \neq \ell_e$
		$\mathsf{com}_i^{[e]} = \text{Com}(\mathsf{seed}_i^{[e]}; \rho_i^{[e]})$
		Rerun the party i
		as the prover to get $[\![t^{[e]}]\!]_i$
		$\Delta r^{[e]} = y - \sum_{i\neq\ell_e} [\![r^{[e]}]\!]$
		$h_e = \text{Hash}_1(\Delta r^{[e]}, \mathsf{com}_1^{[e]}, \dots, \mathsf{com}_n^{[e]})$
		From $\Delta r^{[e]}$, deduce $\Delta t^{[e]}$.
		$[\![t^{[e]}]\!] = t - \Delta t^{[e]} - \sum_{i\neq\ell_e} [\![t^{[e]}]\!]_i$
		$h'_e = \text{Hash}_3(\tilde{x}^{[e]}, [\![t^{[e]}]\!])$
		Check $h = \text{Hash}_2(h_1, \dots, h_M)$
		Check $h' = \text{Hash}_4((h'_e)_{e\in J})$
		Return 1

Protocol 2: Zero-knowledge argument for Subset Sum Problem via MPC-in-the-head with rejection, using cut-and-choose strategy to prove binarity.

Proof size. Let us recall that the couples $(\mathsf{seed}_i, \rho_i)$ are sampled using a tree PRG, sending $(\mathsf{seed}_i^{[e]}, \rho_i^{[e]})_{i \neq \ell_e}$ costs at most $\lambda \cdot \log_2(N)$ bits by iteration. The communication cost (in bits) of the protocol is then

$$\textsc{Size} = 4\lambda + \lambda \cdot \tau \cdot \log_2 \frac{M}{\tau} + \tau \cdot [n \cdot \log_2(A-1) + n + \lambda \log_2 N + 2\lambda] \,.$$

Here again, the obtained size is independent of the modulus q (and of the size of the integers $\{w_j\}, t$).

4.3 Decreasing the Rejection Rate

The two above protocols have a rejection rate around $\tau n/A$ which implies that we must take $A = \Theta(\tau n)$ to obtain a constant (small) rejection rate. In practice, this results in a significant increase in the communication cost. Let us for instance consider Protocol 1 with $(\tau, N, A) = (16, 280, 2^{13})$. For this setting, the proof size is about 15.6 KB for a rejection rate of 0.394. If we increased A to get a rejection rate below 0.003, we should take $A = 2^{21}$ and the proof size would be 23.6 KB.

A better strategy consists in allowing the prover to abort a few of the τ iterations. Let us assume that the verifier accepts the proof if the prover can answer to $\tau - \eta$ challenges among the τ iterations. This slightly increases the soundness error, but it can also significantly decrease the global rejection rate. If we denote p_{rej} the probability that an iteration aborts, then the global rejection rate of this strategy is given by

$$1 - \sum_{i=0}^{\eta} \binom{\tau}{i} \cdot (1 - p_{\text{rej}})^{\tau - i} \cdot p_{\text{rej}}^{i} \,. \tag{3}$$

At the same time, the soundness error for Protocol 1 becomes

$$\sum_{i=0}^{\eta} \binom{\tau}{i} \cdot (1-\epsilon)^i \cdot \epsilon^{\tau - i}$$

where $\epsilon = \frac{1}{N} + \frac{1}{q'} - \frac{1}{q'} \cdot \frac{1}{N}$ is the soundness error of a single iteration. Using this strategy with $\tau = 20$ and $\eta = 3$, the proof size is of 16.7 KB for a rejection rate of 0.003 (instead of 23.6 KB with the naive strategy).

The same strategy also applies to Protocol 2. The rejection rate is also given by Equation (3) while the soundness error becomes

$$\max_{M-\tau \leq k \leq M} \left\{ \frac{\binom{k}{M-\tau}}{\binom{M}{M-\tau}} \cdot \sum_{i=0}^{\eta} \left[\binom{k-M+\tau}{i} \left(1 - \frac{1}{N}\right)^i \left(\frac{1}{N}\right)^{k-M+\tau-i} \right] \right\}.$$

In any case, the prover always answers to at most $\tau - \eta$ challenges of the verifier (even if the prover aborts less than η among the τ iterations) so that the communication cost is roughly that of $\tau - \eta$ iterations. Additionally, for each unanswered challenge, the prover must further send two hash digests to enable

the verifier to recompute and check h and h'. Thus the new proof size (in bits) for Protocol 1 is

$$\begin{aligned}\textsc{Size}_\eta &= 4\lambda + \eta \cdot 4\lambda \\ &\quad + (\tau - \eta) \cdot [n \cdot (\log_2(A-1) + \log_2(q')) + \log_2(q') + \lambda \log_2 N + 2\lambda] \ ,\end{aligned}$$

while the new proof size (in bits) for Protocol 2 is

$$\begin{aligned}\textsc{Size}_\eta = 4\lambda + \eta \cdot 4\lambda + \lambda \cdot \tau \cdot \log_2 \frac{M}{\tau} \\ + (\tau - \eta) \cdot [n \cdot \log_2(A-1) + n + \lambda \log_2 N + 2\lambda] \ .\end{aligned}$$

We note that in practice, given a target security level and a target rejection probability, one needs to use a slightly increased τ (or N) to compensate for the loss in terms of soundness. While this shall slightly increase the proof size, the above approach (with $\eta > 0$) still provides better trade-offs than the original approach ($\eta = 0$).

5 Instantiations and Performances

5.1 Subset Sum Instances

We recall in this section known techniques to solve the modular subset sum problem (SSP) defined by (1). It is well-known that the hardness of an SSP instance depends greatly on its *density* defined as $d = n/\log_2 q$. If the SSP instance is too sparse (e.g. $d < 1/n$) or too dense (e.g. $d > n/\log^2 n$) then the problem can be solved in polynomial time (see e.g. [CJL+92] and references therein). We shall therefore only consider SSP instances with density $d \simeq 1$ (i.e. $q \simeq 2^n$) which are arguably the hardest ones [IN96].

In this case, simple algorithms exist based on brute force enumeration at $O(2^n)$ time and constant space, or time-space tradeoff [HS74] with $O(2^{n/2})$ time and space complexities. The first non-trivial algorithm was published by Schroeppel and Shamir [SS81] with time complexity $O(2^{n/2})$ and space complexity $O(2^{n/4})$. Later, faster algorithms were proposed with similar time and space complexities, e.g. $\tilde{O}(2^{0.337n})$ by Howgrave-Graham and Joux [HJ10]and $\tilde{O}(2^{0.283n})$ by Bonnetain, Bricout, Schrottenloher and Shen [BBSS20]. The latter algorithms neglect the cost to access an exponential memory but even with this optimistic assumption, for $n = 256$, all known algorithms require at least a time complexity lower-bounded by 2^{128} operations or memory of size at least 2^{72} bits. There also exists a vast literature on quantum algorithms for solving the SSP (see [BBSS20] and references therein). The best (heuristic) quantum complexity from [BBSS20] has time complexity $\tilde{O}(2^{0.216n})$ and thus requires about 2^{64} quantum operations and quantum memory for $n = 256$. In the following, we, therefore, consider the efficiency of our protocols for $n = 256$.

5.2 Zero Knowledge Protocols

Let us consider the subset sum problem with $n = 256$. We propose in Table 1 several sets of parameters for our two protocols which target a security of 128 bits. We provide two kinds of instantiations to give the reader an idea of the obtained performance while changing the number of parties. The first ones correspond to instantiations with fast computation. The second ones correspond to instantiations that achieve smaller communication costs but slower computation. For each setting, we suggest two parameter sets: one achieving a rejection rate around 0.4 and the other one achieving a rejection rate between 0.001 and 0.004.

Protocol	Parameters					Proof size	Rej. rate	Soundness err.
	τ	η	N	A	M			
Shamir [Sha86]	219	-	-	-	-	1186 KB	-	128 bits
[LNSW13]	219	-	-	-	-	2350 KB	-	128 bits
Beullens [Beu20]	14	-	1024	-	4040	122 KB	-	128 bits
Protocol 1 (batching)	26	0	32	2^{14}	-	25.7 KB	0.334	130 bits
Protocol 1 (batching)	31	3	32	2^{14}	-	27.9 KB	0.001	128 bits
Protocol 2 (C&C)	27	0	32	2^{14}	462	17.4 KB	0.344	128 bits
Protocol 2 (C&C)	33	3	32	2^{14}	470	19.6 KB	0.002	128 bits
Protocol 1 (batching)	17	0	256	2^{13}	-	16.6 KB	0.412	135 bits
Protocol 1 (batching)	21	3	256	2^{13}	-	17.7 KB	0.004	133 bits
Protocol 2 (C&C)	19	0	256	2^{13}	954	13.0 KB	0.448	128 bits
Protocol 2 (C&C)	24	3	256	2^{14}	952	15.4 KB	0.001	128 bits

Table 1: Comparison of state-of-the-art zero-knowledge protocols for proving the knowledge of an SSP instance (with $n = 256$ and $q \approx 2^{256}$).

We provide in Table 1 the performance of the other zero-knowledge protocols proving the knowledge of an SSP solution. The only other protocol designed for the subset sum problem is Shamir's one [Sha86]. We can also compare these protocols with [LNSW13] which is an adaptation of Stern's protocol to the ISIS (inhomogeneous short integer solution) problem. The remaining articles in the literature about proofs for the ISIS problem are restricted to the case where the modulus q is prime. We add Beullens' protocol [Beu20] for ISIS with prime q to the comparison.

We provide in the full version [FMRV22] the performances of the obtained signatures when applying the Fiat-Shamir transform [FS87] to our protocols.

5.3 Comparison with Generic Techniques

In this section, we compare our scheme with efficient generic techniques to prove the knowledge of an SSP solution. Among those techniques, we consider SNARKs

(e.g. [Gro16]), "compressed" proof systems such as *Bulletproofs* [BBB+18] and STARKs (e.g. [BBHR18]). For the sake of accuracy, we split the notation for the security level of the subset sum instance, denoted κ, and of the zero-knowledge argument, denoted λ. Adapting the analysis of Section 3.5 to this setting, we get a communication cost of $\Theta(\lambda^2 + \lambda \cdot \kappa)$ for our protocols.

The asymptotic size of [Gro16] arguments is roughly[5] $\Omega(\lambda^3)$ which is asymptotically larger than ours, but for $\kappa = \lambda = 128$, these arguments will be shorter than ours (within the range of 700-800 bytes). Using *Bulletproofs* [BBB+18], one can obtain an asymptotic communication cost of $\Omega(\log(\kappa)(\lambda+\kappa))$, and about 600 bytes for $\kappa = \lambda = 128$. Although SNARKs and "compressed" proof systems give shorter arguments than ours for $\kappa = \lambda = 128$, they both require stronger and non post-quantum computational assumptions. In particular, [Gro16] requires a trusted setup and a non-falsifiable assumption, while Bulletproofs rely on the algebraic group model in their non-interactive version [GOP+21]. In comparison, the security of our arguments only relies on weak post-quantum assumptions (PRG, collision-resistant hash functions).

Regarding STARKs [BBHR18], their security assumptions are similar to ours. When applying STARKs to the subset sum problem, one gets arguments of size $\Omega(\lambda^2 \cdot \log^2 \kappa)$, which is larger than ours.[6]

6 Further Applications

As illustrated on the subset sum problem, our technique of sharing over the integers with rejection is –more generally– instrumental to a context of a secret vector $s \in \mathbb{Z}_q^n$ with small coefficients. Since the communication cost of our protocols is independent of the size q of the ring $\mathbb{Z}_q$, the gain in communication is higher when the modulus q is high. But it does not need to have a modulus as high as in the subset sum problem to be interesting. In the three subsections, we present the performance of our schemes with the sharing over the integers on three other applications with moderate-size modulus:

- to prove the knowledge of a solution of an ISIS problem instance,
- to prove the knowledge of a secret key and plaintext(s) matching a (set of) FHE ciphertext(s),
- to construct an efficient digital signature based on Boneh-Halevi-Howgrave-Graham pseudo-random function.

Another advantage of the sharing on the integers is that we can perform any operation on it with any modulus. We used this property in one of our protocols to check multiplication triples in a smaller field. This property can be also useful when we want to prove that the same secret vector verifies many relations using distinct modulus.

[5] This is due to sub-exponential attacks on the discrete logarithm in the target group which also impacts the size of elements of the second group of the bilinear structure.

[6] The λ^2 factor is obtained by λ for the hash digest size times λ for the number of evaluation points in the FRI protocol (which scales with the soundness error). The $\log^2 \kappa$ factor comes from the size κ of the program verifying the SSP instance.

6.1 Short Integer Solution Problem

Given a matrix $A \in \mathbb{Z}^{m \times n}$ and a vector $u \in \mathbb{Z}_m$, the inhomogenous short integer solution (ISIS) problem consists in finding a vector $s \in \mathbb{Z}^n$ with small coefficients such that

$$As = u \bmod q.$$

The Ling-Nguyen-Stehlé-Wang protocol [LNSW13], which is an adaptation of Stern's protocol, has been for a long time the only zero-knowledge exact protocol which proves the knowledge of a solution of an ISIS instance. Other protocols existed but they were only relaxed proofs, *i.e.* they prove the knowledge of an s' and c satisfying $As' = cu \bmod q$. These protocols can be useful in some contexts, but they are not suited to prove the exact statement.

Recently, new exact proofs [BLS19,ENS20,LNS21,BN20,Beu20] have been published. However, all these new protocols require an assumption on the modulus q to work: some of them only require that q is a prime number when the others require that q is an NTT-friendly prime number. In the state of the art, the only protocol which works for any q (even when q is not a prime) is [LNSW13].

We can adapt our protocols of Section 4 to the case of the ISIS problem. The linear constraint "$As = u$" is free in communication as it was the case for "$t = \langle w, x \rangle$" for the subset sum problem (see Section 3.2). The hard part is to prove that the secret s satisfies $\|s\|_\infty \leq \beta$ for some bound β. To proceed, we decompose s as $k := \lceil \log_2(2\beta + 1) \rceil$ vectors $(s_0, \ldots, s_{k-1})$ of $\{0,1\}^n$ such that

$$s = \sum_{i=0}^{k-2} 2^i s_i + (2\beta - 2^{k-1} + 1) s_{k-1} - \beta \ . \tag{4}$$

If all vectors s_i belong to $\{0,1\}$, the above relation gives that $\|s\|_\infty \leq \beta$. So we just need to give the sharing $\{[\![s_i]\!]\}_{i \in \{0,\ldots,k-1\}}$ to the MPC protocol instead of $[\![s]\!]$. The latter can then check that $\{[\![s_i]\!]\}_{i \in \{0,\ldots,k-1\}}$ are binary vectors and that $A[\![s]\!]$ corresponds to u modulo q where $[\![s]\!]$ is recovered by linearity of Equation (4). The proof sizes of the resulting protocols are given by the formulae as before, we just need to consider that the length of the secret is $n \cdot k$ (instead of n).

We compare our protocols with the state of the art in Table 2 on the two following ISIS problems:

1. $\|s\|_\infty \leq 1$, $m = 1024$, $n = 2048$, $q \approx 2^{32}$
2. Binary s, $m = 512$, $n = 4096$, $q \approx 2^{61}$

For both instances, we have $k \cdot n = 4096$. For our protocols, we choose the following parameters:

- Protocol 1 (batch product verification):

$$A = 2^{16},\ N = 128,\ q' \approx A,\ \tau = 23,\ \eta = 3.$$

- Protocol 2 (cut-and-choose strategy):

$$A = 2^{16},\ N = 256,\ q' \approx A,\ M = 952,\ \tau = 24,\ \eta = 3.$$

We can remark that our protocols have the same communication cost for both instances. It comes from the fact that their proof size is independent of the modulus q. Even when q is prime (and larger than 2^{32}), our Protocol 2 (with the cut-and-choose phase) has smaller communication cost than Beullens' protocol and this while taking less aggressive parameters towards size against speed (the parameters used in [Beu20] are $(\tau, M, N) = (14, 4040, 2^{10})$). We also observe that our protocols achieve proof sizes which are more than 10 times smaller than those of [LNSW13], the only previous protocol supporting any modulus q.

Protocol	Year	Any q	Instance 1		Instance 2	
			Proof Size	Rej. Rate	Proof Size	Rej. Rate
[LNSW13]	2013	✓	3600 KB	-	8988 KB	-
[BN20]	2020	q prime	-	-	4077 KB	-
[Beu20]	2020	q prime	233 KB	-	444 KB	-
Our Protocol 1	2022	✓	291 KB	0.04	291 KB	0.04
Our Protocol 2	2022	✓	184 KB	0.05	184 KB	0.05
[BLS19]	2019	q prime + NTT	384 KB	0.92		
[ENS20]	2020	q prime + NTT	47 KB	0.95		
[LNS21]	2021	q prime + NTT	33.3 KB	0.85		
Aurora [BCR+19]	2019	q prime + NTT	71 KB	-		
Ligero [AHIV17]	2017	q prime + NTT	157 KB	-		

Table 2: Comparison with the existing exact protocols which prove the knowledge of the solution of a ISIS instance.

6.2 Fully Homomorphic Encryption

Our zero-knowledge protocols also find application to fully homomorphic encryption (FHE). We can indeed adapt our protocols to prove the knowledge of a secret key matching a (set of) FHE-encrypted plaintext(s). We elaborate on this application hereafter for the particular case of TFHE (Torus FHE) [CGGI20] which is currently one of the FHE schemes with the best performances in practice.

For some $q \in \mathbb{N}$, let $\mathbb{T}_q = q^{-1}\mathbb{Z}/\mathbb{Z}$ be the discretized torus with q elements, i.e. the submodule of the real torus with representative $\{i/q \ ; \ i \in \mathbb{Z}_q\}$ [Joy21]. In practice, q is often chosen to be 2^{32} or 2^{64} in order to match the word-size and arithmetic operations of common CPUs. For this reason, we shall consider that q is a power of 2 in the following (although the described application can be easily generalized to any q). TFHE relies on so called TLWE (Torus Learning With Error) encryption. Let $p \mid q$ and $\delta = q/p$. The plaintext space is defined as $\mathbb{Z}_p$ while the key space is defined as $\{0,1\}^n \subset \mathbb{Z}^n$. Let $s = (s_1, \ldots, s_n) \in \{0,1\}^n$ be a secret key. The TLWE encryption of a plaintext $\mu \in \mathbb{Z}_p$ under the secret key s and with *error* $e \in \mathbb{Z}$ is defined as

$$c = (a_1, \ldots, a_n, b) \in \mathbb{T}_q^{n+1} \quad \text{where} \quad \begin{cases} \mu^* = \frac{\delta\mu + e \bmod q}{q} \in \mathbb{T}_q \\ b = \sum_{j=1}^{n} s_j \cdot a_j + \mu^* \end{cases}$$

The a_i's are random elements of $\mathbb{T}_q$ which are sampled at encryption time or which arise from the homomorphic operations between other ciphertexts. The value $e \in \mathbb{Z}$ is the error which must satisfies $|e| < \delta/2$ to ensure the correctness of the decryption.

Proving the knowledge of a key s and plaintext μ for which $c = (a_1, \ldots, a_n, b)$ is a correct TLWE encryption of μ under s can be achieved by proving the knowledge of a binary vector

$$x = (s_1, \ldots, s_n) \mid (\mu_1, \ldots, \mu_{\ell_p}) \mid (e_1, \ldots, e_{\ell_e})$$

where $\ell_p = \log_2 p$ and ℓ_e is such that $e \in \{-2^{\ell_e - 1}, \ldots, 2^{\ell_e - 1} - 1\}$, and which satisfies

$$\sum_{i=1}^{n} \bar{a}_i s_i + \sum_{i=1}^{\ell_p} (2^{i-1}\delta)\mu_i + \sum_{i=1}^{\ell_e} (2^{i-1}) e_i = \bar{b} + 2^{\ell_e - 1} \pmod{q}$$

where $\bar{a}_i \in \mathbb{Z}$ (resp. $\bar{b} \in \mathbb{Z}$) is the integer such that $a_i = \bar{a}_i/q \in \mathbb{T}_q$ (resp. $b = \bar{b}/q \in \mathbb{T}_q$) and where the error is $e := -2^{\ell_e - 1} + \sum_{i=1}^{\ell_e} (2^{i-1}) e_i$. The application of our protocols to this context is immediate. We note that the secret binary vector is of size $n' = n + \ell_p + \ell_e$ when the underlying plaintext must remain secret while it is of size $n' = n + \ell_e$ if the plaintext is public. In the latter case, the value of the sum is $t = \bar{b} + 2^{\ell_e - 1} - \mu$. We can also use our protocols to prove the knowledge of a secret key and a set of plaintexts matching a set of ciphertexts. For m ciphertexts, we obtain m linear relations with a binary vector of size $n' = n + m \cdot (\ell_p + \ell_e)$ (or $n' = n + m \cdot \ell_e$ in the public plaintext setting).

Remark 2. Proving the knowledge of a single key-plaintext pair matching a given ciphertext might not be relevant on its own. Indeed, for the typical parameters given above, the obtained SSP instance might not be hard (*i.e.* finding *a* solution is not hard while finding the original key-plaintext pair is still hard). However, such proof is still useful whenever proving additional properties involving the underlying secret key and/or plaintext. In such contexts, finding a solution to the SSP instance which does not match the original key-plaintext pair is useless.

According to [Joy21], typical parameters for a TLWE encryption are $q = 2^{32}$ or $q = 2^{64}$ and $n = 630$. Depending on the exact message space and error space, we have $n' \in (n, n + \log_2 q]$. Table 3 gives the obtained communication cost for proving the knowledge of the key (and plaintexts) corresponding to 1, 64 and 1024 TLWE ciphertexts using our protocols (assuming $q = 2^{64}$ and $\ell_e + \ell_p = 64$). For the sake of comparison, we also give the communication obtained with Shamir's protocol [Sha86]. We note that the latter and the LNSW protocol [LNSW13] are the only previous protocols which can work with such values of q and the LNSW protocol is always heavier than Shamir's in this context. We observe that our protocols always gain more than a factor 10 (for Protocol 1) and 20 (for Protocol 2) for the obtained communication cost compared to Shamir's protocol.

Protocol	Parameters					Proof size	Rej. rate	Soundness err.
	τ	η	N	A	M			
1 ciphertext								
Shamir [Sha86]	219	-	-	-	-	845 KB	-	128 bits
Protocol 1 (batching)	19	2	256	2^{15}	-	46.1 KB	0.007	128 bits
Protocol 2 (C&C)	24	3	256	2^{15}	952	34.0 KB	0.002	128 bits
64 ciphertexts								
Shamir [Sha86]	219	-	-	-	-	8.48 MB	-	128 bits
Protocol 1 (batching)	19	2	256	2^{18}	-	356 KB	0.005	129 bits
Protocol 2 (C&C)	24	3	256	2^{18}	952	236 KB	0.001	128 bits
1024 ciphertexts								
Shamir [Sha86]	219	-	-	-	-	77.9 MB	-	128 bits
Protocol 1 (batching)	19	2	256	2^{22}	-	5.90 MB	0.003	129 bits
Protocol 2 (C&C)	24	3	256	2^{21}	952	3.65 MB	0.006	128 bits

Table 3: Comparison of ZK protocols for TFHE decryption.

6.3 Digital Signatures from Boneh-Halevi-Howgrave-Graham PRF

As another application, we present a short and efficient candidate post-quantum signature scheme based on an elegant pseudo-random function (PRF) proposed by Boneh, Halevi, and Howgrave-Graham in 2001 [BHH01].

Let p be a public m-bit prime number that defines the PRF message space as $\mathbb{Z}_p$. A secret key for the PRF is an element $x \in \mathbb{Z}_p$ picked uniformly at random. We denote $\mathrm{MSB}_\delta(t)$ the δm most significant bits of an m-bit element $t \in \mathbb{Z}_p$.[7] The value of the PRF on the message $m \in \mathbb{Z}_p$ for the secret-key $x \in \mathbb{Z}_p$ is $F_x(m) = \mathrm{MSB}_\delta((x+m)^{-1} \bmod p)$.

Our signature scheme follows the blueprint of most signatures based on the MPCitH paradigm since the proposal of Picnic [CDG+17]: the public key is made of the outputs of Boneh *et al.*'s PRF on t public messages in $\{1, \ldots, t\}$, i.e. the δm-bit elements $y_1, \ldots, y_t$ such that

$$y_i := \mathrm{MSB}_\delta((x+i)^{-1} \bmod p) \text{ for } i \in \{1, \ldots, t\}$$

and the signature consists of a non-interactive proof of knowledge of $x, z_1, \ldots, z_t$ (parametrized by the signed message using the Fiat-Shamir heuristic) such that

$$(x+1)(2^{(1-\delta)m} y_1 + z_1) \equiv \cdots \equiv (x+t)(2^{(1-\delta)m} y_t + z_t) \equiv 1 \bmod p \quad (5)$$

$$\text{and} \qquad z_1, \ldots, z_t \in \{0, \ldots, 2^{(1-\delta)m} - 1\} \quad (6)$$

where $z_1, \ldots, z_t$ are the $(1-\delta)m$ least significant bits of $(x+1)^{-1} \bmod p$, ..., $(x+t)^{-1} \bmod p$. Note that the condition (6) on the size of the z_i's is fundamental since otherwise, it is easy for an attacker to find a witness.

[7] We assume hereafter that $\delta m \in \mathbb{Z}$. Otherwise, one should take the nearest integer $\lfloor \delta m \rceil$ instead.

In our applications, the values of t and δ are chosen to prevent all known classical attacks and target a 128-bit security level.

Let's fix t, the number of outputs of the PRF. Then, to ensure that the equations (5) and (6) have a unique witness, we add the constraint $\delta \geq 1/t$ so that the t PRF outputs define (heuristically) the secret x uniquely. To avoid brute-force attacks from a single output of the PRF, the hidden most significant bits of one output should be at least 128 bits, thus $\log p \geq \frac{128}{1-\delta}$. Otherwise, an attacker could reconstruct a possible key for the PRF and then evaluate the other outputs with this candidate to test it.

It is possible to apply generically the MPCitH paradigm to prove (5) and (6), but proving (6) seems inefficient (e.g. by using a binary decomposition and proving consistency). Instead, we can use our secret sharing over the integers for proving the knowledge of small z_i's by sharing them as a sum of "small" integers which directly proves that the z_i's are indeed small.

Proving Equation (5). Instead of proving the t products of (5) separately, the prover can batch them into a linear combination where coefficients $\gamma_1, ..., \gamma_t$ are provided by the verifier, *i.e.* the prover proves the equation

$$\sum_{i=1}^{t} \gamma_i \cdot \Big((x+i)(2^{(1-\delta)m} y_i + z_i) - 1 \Big) = 0 \bmod p,$$

or equivalently,

$$x \cdot \left(\sum_{i=1}^{t} \gamma_i z_i \right) = - \sum_{i=1}^{t} \gamma_i \left(x \cdot 2^{(1-\delta)m} y_i + i \cdot 2^{(1-\delta)m} y_i + i \cdot z_i - 1 \right) \bmod p. \quad (7)$$

If one of the products is not equal to 1 in (5), then (7) is satisfied only with a probability of $\frac{1}{p}$. And to prove (7), one can use the protocol of [BN20] with a single multiplication on $\mathbb{Z}_p$ (for the left-hand side of (7), the right-hand side being a linear combination of the witness). The resulting MPC protocol produces false positives with probability $1/p + (1 - 1/p) \cdot 1/p := 2/p + 1/p^2$, and thus the obtained zero-knowledge argument has a soundness error of

$$\epsilon = \frac{1}{N} + \left(1 - \frac{1}{N}\right) \left(\frac{2}{p} + \frac{1}{p^2} \right).$$

Proving Equation (6). It remains to prove that z_i is in $\{0, \ldots, B-1\}$ with $B = 2^{(1-\delta)m}$ in (6) for $i \in \{1, \ldots, t\}$. To share z_i, we use our secret sharing over the integers of Section 3.2. Since the z_i are not binary but in a larger range, we need to adapt the rejection rules. Following exactly the same reasoning as in Section 3.2, we get that the prover must abort if there exists an index $j \in [t]$ for which $z_j - [\![z_j]\!]_{i^*} \geq 1$ or $z_j - [\![z_j]\!]_{i^*} \leq -A + B - 1$. The resulting rejection rate is given by

$$p_{\text{rej}} = 1 - \left(1 - \frac{B-1}{A}\right)^{t \cdot \tau} \approx t \cdot \tau \cdot \frac{B-1}{A}.$$

Even without proving anything on the range of z_j, the verifier knows that

$$\forall j \in [t], -A + B \leq z_j \leq A - 1$$

thanks to (2) (generalized). In practice, we settle this range, implying that there is a slack between the underlying hard problem and the proven statement. A malicious prover can use bigger values for z_i, and this is equivalent to ignoring some bits of y_i. A malicious prover can ignore up to $\log_2 \frac{A}{B} \approx \log_2 \frac{t \cdot \tau}{p_{\text{rej}}}$ bits for each PRF output, and thus it reduces the security of $t \cdot \log_2 \frac{t \cdot \tau}{p_{\text{rej}}}$ bits. A way to fix this security loss without increasing the size of p (and of the key) is to reveal a few more PRF outputs to guarantee that the key is still heuristically unique. In theory, this decreases the security but for state-of-the-art algorithms, this stays beyond the capacity of the best-known algorithms for small t. In fact, we need to reveal $\tilde{t} \geq t$ outputs of the PRF such that

$$\tilde{t} \cdot \delta \cdot m - \tilde{t} \cdot \log \left(\frac{\tilde{t} \cdot \tau}{p_{\text{rej}}} \right) > m.$$

In other words, since $\delta \geq \frac{1}{t}$, we adapt this constraint as

$$\delta \geq \frac{1}{\tilde{t}} + \frac{1}{m} \log_2 \left(\frac{\tilde{t} \cdot \tau}{p_{\text{rej}}} \right).$$

This leads to a scheme (formally described in the full version [FMRV22]) with the communication cost (in bits):

$$4\lambda + \tau \cdot (\log_2 p + \tilde{t} \cdot \log_2 A + \log_2 p + \log_2 p + \lambda \cdot \log_2 N + 2\lambda),$$

with soundness error (if interactive)

$$\epsilon = \frac{1}{N} + \left(1 - \frac{1}{N}\right) \left(\frac{2}{p} + \frac{1}{p^2}\right),$$

and with forgery security (if non-interactive)

$$\text{cost}_{\text{forge}} = \min_{\tau_1, \tau_2 : \tau_1 + \tau_2 = \tau} \left\{ \frac{1}{\sum_{i=\tau_1}^{\tau} \binom{\tau}{i} p'^i (1-p')^{\tau - i}} + N^{\tau_2} \right\},$$

with $p' := 2/p + 1/p^2$.

We propose in Table 4 some parameters which target 128-bit security (based on the hardness of the so-called *modular inverse hidden number problem*) according to the current cryptanalysis state-of-the-art for Boneh *et al.*'s PRF. We can remark that the achieved signature sizes are competitive with Rainier scheme [DKR+21] (which can produce signatures that are around 5 KB in size too) and outperform all the other signatures based on MPC-in-the-Head paradigm (Picnic4 [KZ22], PorcRoast [Bd20], SDitH [FJR22], ...).

Parameters							Size	p_{rej}
$p \approx 2^m$	$\tilde{t}$	δ	B	A	N	τ		
$\approx 2^{229}$	3	88/229	2^{141}	2^{141+12}	256	16	4 916 B	0.012
$\approx 2^{186}$	4	58/186	2^{128}	2^{128+12}	256	16	4 860 B	0.016
$\approx 2^{175}$	5	47/175	2^{128}	2^{128+12}	256	16	5 074 B	0.019

Table 4: Parameter sets and achieved performances of the signature based on Boneh *et al.*'s PRF, for a 128-bit security.

Regarding the cryptanalysis, the security of Boneh *et al.*'s PRF has been extensively analyzed since 20 years [BHH01,LSSW12,BVZ12,XSH+19] and relies strongly on δ and the number of known PRF outputs. In [XSH+19], Xu, Sarkar, Hu, Wang, and Pan presented a heuristic attack based on Coppersmith's method that breaks Boneh *et al.*'s PRF (for a sufficiently large modulus p) if the number of outputs of the PRF is large enough (depending on δ). However, this polynomial-time attack is not practical and hides *galactic* constant factors. For instance, for $\delta = 1/3$, this attack requires 45 outputs of the PRF and uses a lattice of dimension 209899 in Coppersmith's method.

The best known lattice-based attacks with a small number of PRF outputs are described in [BHH01,BVZ12] and require larger δ's than the ones we use. In order to mount them, an adversary has thus to perform an exhaustive search on the missing bits on several outputs. A precise security analysis is given in the full version of the paper [FMRV22]. For all parameters provided in Table 4 an exhaustive search on (at least) 128 bits has to be performed by the adversary in order to run the attacks from [BHH01,BVZ12].

To the best of our knowledge, the quantum security of Boneh *et al.*'s PRF has not been analyzed yet. Our signature protocol is thus a post-quantum candidate and requires further analysis of its security by quantum algorithm specialists.

Acknowledgements. The authors are supported in part by the French ANR SANGRIA project (ANR-21-CE39-0006). The authors would like to thank Charles Bouillaguet for suggesting investigation of zero-knowledge proofs for the subset sum problem.

References

AHIV17. S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, eds, *ACM CCS 2017*, p. 2087–2104. ACM Press, 2017.

BBB+18. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018*

IEEE Symposium on Security and Privacy, p. 315–334. IEEE Computer Society Press, 2018.

BBHR18. E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018.

BBSS20. X. Bonnetain, R. Bricout, A. Schrottenloher, and Y. Shen. Improved classical and quantum algorithms for subset-sum. In S. Moriai and H. Wang, eds, *ASIACRYPT 2020, Part II*, vol. 12492 of *LNCS*, p. 633–666. Springer, Heidelberg, 2020.

BCR+19. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. In Y. Ishai and V. Rijmen, eds, *EUROCRYPT 2019, Part I*, vol. 11476 of *LNCS*, p. 103–128. Springer, Heidelberg, 2019.

BD10. R. Bendlin and I. Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In D. Micciancio, ed., *TCC 2010*, vol. 5978 of *LNCS*, p. 201–218. Springer, Heidelberg, 2010.

Bd20. W. Beullens and C. de Saint Guilhem. LegRoast: Efficient post-quantum signatures from the Legendre PRF. In J. Ding and J.-P. Tillich, eds, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, p. 130–150. Springer, Heidelberg, 2020.

BDLN16. C. Baum, I. Damgård, K. G. Larsen, and M. Nielsen. How to prove knowledge of small secrets. In M. Robshaw and J. Katz, eds, *CRYPTO 2016, Part III*, vol. 9816 of *LNCS*, p. 478–498. Springer, Heidelberg, 2016.

Beu20. W. Beullens. Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In A. Canteaut and Y. Ishai, eds, *EUROCRYPT 2020, Part III*, vol. 12107 of *LNCS*, p. 183–211. Springer, Heidelberg, 2020.

BGKW90. M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Efficient identification schemes using two prover interactive proofs. In G. Brassard, ed., *CRYPTO'89*, vol. 435 of *LNCS*, p. 498–506. Springer, Heidelberg, 1990.

BHH01. D. Boneh, S. Halevi, and N. Howgrave-Graham. The modular inversion hidden number problem. In C. Boyd, ed., *ASIACRYPT 2001*, vol. 2248 of *LNCS*, p. 36–51. Springer, Heidelberg, 2001.

BLS19. J. Bootle, V. Lyubashevsky, and G. Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In A. Boldyreva and D. Micciancio, eds, *CRYPTO 2019, Part I*, vol. 11692 of *LNCS*, p. 176–202. Springer, Heidelberg, 2019.

BN20. C. Baum and A. Nof. Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, eds, *PKC 2020, Part I*, vol. 12110 of *LNCS*, p. 495–526. Springer, Heidelberg, 2020.

BVZ12. A. Bauer, D. Vergnaud, and J.-C. Zapalowicz. Inferring sequences produced by nonlinear pseudorandom number generators using Coppersmith's methods. In M. Fischlin, J. Buchmann, and M. Manulis, eds, *PKC 2012*, vol. 7293 of *LNCS*, p. 609–626. Springer, Heidelberg, 2012.

CDG+17. M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, eds, *ACM CCS 2017*, p. 1825–1842. ACM Press, 2017.

CGGI20. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

CGH00. D. Catalano, R. Gennaro, and S. Halevi. Computing inverses over a shared secret modulus. In B. Preneel, ed., *EUROCRYPT 2000*, vol. 1807 of *LNCS*, p. 190–206. Springer, Heidelberg, 2000.

CJL+92. M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C. Schnorr, and J. Stern. Improved low-density subset sum algorithms. *Comput. Complex.*, 2:111–128, 1992.

DKR+21. C. Dobraunig, D. Kales, C. Rechberger, M. Schofnegger, and G. Zaverucha. Shorter signatures based on tailor-made minimalist symmetric-key crypto. Cryptology ePrint Archive, Report 2021/692, 2021.

ENS20. M. F. Esgin, N. K. Nguyen, and G. Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In S. Moriai and H. Wang, eds, *ASIACRYPT 2020, Part II*, vol. 12492 of *LNCS*, p. 259–288. Springer, Heidelberg, 2020.

FJR22. T. Feneuil, A. Joux, and M. Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. Cryptology ePrint Archive, Report 2022/188, 2022.

FMRV22. T. Feneuil, J. Maire, M. Rivain, and D. Vergnaud. Zero-knowledge protocols for the subset sum problem from MPC-in-the-head with rejection. Cryptology ePrint Archive, Report 2022/223, 2022.

FS87. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, ed., *CRYPTO'86*, vol. 263 of *LNCS*, p. 186–194. Springer, Heidelberg, 1987.

GMO16. I. Giacomelli, J. Madsen, and C. Orlandi. Zkboo: Faster zero-knowledge for boolean circuits. In T. Holz and S. Savage, eds, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, p. 1069–1083. USENIX Association, 2016.

GMR89. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

GOP+21. C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat–shamir bulletproofs are non-malleable (in the algebraic group model). Cryptology ePrint Archive, Report 2021/1393, 2021.

Gro16. J. Groth. On the size of pairing-based non-interactive arguments. In M. Fischlin and J.-S. Coron, eds, *EUROCRYPT 2016, Part II*, vol. 9666 of *LNCS*, p. 305–326. Springer, Heidelberg, 2016.

HJ10. N. Howgrave-Graham and A. Joux. New generic algorithms for hard knapsacks. In H. Gilbert, ed., *EUROCRYPT 2010*, vol. 6110 of *LNCS*, p. 235–256. Springer, Heidelberg, 2010.

HS74. E. Horowitz and S. Sahni. Computing partitions with applications to the knapsack problem. *J. ACM*, 21(2):277–292, 1974.

IKOS09. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

IN96. R. Impagliazzo and M. Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, 1996.

Joy21. M. Joye. Guide to fully homomorphic encryption over the [discretized] torus. Cryptology ePrint Archive, Report 2021/1402, 2021.

Kar72. R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, eds, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, p. 85–103. Plenum Press, New York, 1972.

KKW18. J. Katz, V. Kolesnikov, and X. Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In D. Lie, M. Mannan, M. Backes, and X. Wang, eds, *ACM CCS 2018*, p. 525–537. ACM Press, 2018.

KZ22. D. Kales and G. Zaverucha. Efficient lifting for shorter zero-knowledge proofs and post-quantum signatures. Cryptology ePrint Archive, Paper 2022/588, 2022.

LN17. Y. Lindell and A. Nof. A framework for constructing fast MPC over arithmetic circuits with malicious adversaries and an honest-majority. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, eds, *ACM CCS 2017*, p. 259–276. ACM Press, 2017.

LNS21. V. Lyubashevsky, N. K. Nguyen, and G. Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In J. Garay, ed., *PKC 2021, Part I*, vol. 12710 of *LNCS*, p. 215–241. Springer, Heidelberg, 2021.

LNSW13. S. Ling, K. Nguyen, D. Stehlé, and H. Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In K. Kurosawa and G. Hanaoka, eds, *PKC 2013*, vol. 7778 of *LNCS*, p. 107–124. Springer, Heidelberg, 2013.

LPS10. V. Lyubashevsky, A. Palacio, and G. Segev. Public-key cryptographic primitives provably as secure as subset sum. In D. Micciancio, ed., *TCC 2010*, vol. 5978 of *LNCS*, p. 382–400. Springer, Heidelberg, 2010.

LSSW12. S. Ling, I. E. Shparlinski, R. Steinfeld, and H. Wang. On the modular inversion hidden number problem. *J. Symb. Comput.*, 47(4):358–367, 2012.

Lyu09. V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In M. Matsui, ed., *ASIACRYPT 2009*, vol. 5912 of *LNCS*, p. 598–616. Springer, Heidelberg, 2009.

MH78. R. C. Merkle and M. E. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Trans. Inf. Theory*, 24(5):525–530, 1978.

Odl90. A. M. Odlyzko. The rise and fall of knapsack cryptosystems. Cryptology and computational number theory, Lect. Notes AMS Short Course, Boulder/CO (USA) 1989, Proc. Symp. Appl. Math. 42, 75-88 (1990)., 1990.

Reg05. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, eds, *37th ACM STOC*, p. 84–93. ACM Press, 2005.

Sha86. A. Shamir. A zero-knowledge proof for knapsacks. presented at a workshop on Probabilistic Algorithms, Marseille, 1986.

SS81. R. Schroeppel and A. Shamir. A T=O($2^{n/2}$), S=O($2^{n/4}$) algorithm for certain NP-complete problems. *SIAM J. Comput.*, 10(3):456–464, 1981.

Ste94. J. Stern. A new identification scheme based on syndrome decoding. In D. R. Stinson, ed., *CRYPTO'93*, vol. 773 of *LNCS*, p. 13–21. Springer, Heidelberg, 1994.

XSH+19. J. Xu, S. Sarkar, L. Hu, H. Wang, and Y. Pan. New results on modular inversion hidden number problem and inversive congruential generator. In A. Boldyreva and D. Micciancio, eds, *CRYPTO 2019, Part I*, vol. 11692 of *LNCS*, p. 297–321. Springer, Heidelberg, 2019.