Privacy and Integrity Preserving Computations with CRISP

Sylvain Chatel *EPFL*
Apostolos Pyrgelis *EPFL*
Juan Ramón Troncoso-Pastoriza *EPFL*
Jean-Pierre Hubaux *EPFL*

Abstract

In the digital era, users share their personal data with service providers to obtain some utility, e.g., access to high-quality services. Yet, the induced information flows raise privacy and integrity concerns. Consequently, cautious users may want to protect their privacy by minimizing the amount of information they disclose to curious service providers. Service providers are interested in verifying the integrity of the users' data to improve their services and obtain useful knowledge for their business. In this work, we present a generic solution to the trade-off between privacy, integrity, and utility, by achieving authenticity verification of data that has been encrypted for offloading to service providers. Based on lattice-based homomorphic encryption and commitments, as well as zero-knowledge proofs, our construction enables a service provider to process and reuse third-party signed data in a privacy-friendly manner with integrity guarantees. We evaluate our solution on different use cases such as smart-metering, disease susceptibility, and location-based activity tracking, thus showing its versatility. Our solution achieves broad generality, quantum-resistance, and relaxes some assumptions of state-of-the-art solutions without affecting performance.

1 Introduction

In our inter-connected world, people share personal information collected from various entities, networks, and ubiquitous devices (i.e., data sources) with a variety of service providers, in order to obtain access to services and applications. Such data flows, which typically involve a user, a data source, and a service provider (as depicted in Figure 1), are common for a wide range of use cases, e.g., smart metering, personalized health, location-based activity tracking, dynamic road tolling, business auditing, loyalty programs, and pay-as-you-drive insurance. However, due to conflicting interests of the involved parties, such data interactions inherently introduce a trade-off between *privacy*, *integrity*, and *utility*.

Some users seek to protect their *privacy* by minimizing the amount of personal information that they disclose to *curious* third-parties. Service providers are interested in maintaining the value obtained from the users' data. To this end, service providers are concerned about verifying the *integrity* of the data shared by their users, i.e., ensure that the user's data has been certified by a trusted, external, data source. Both parties want to obtain some *utility* from these data flows: Service providers want to use the data for various computations that yield useful knowledge for their business or services, and users share part of their data to obtain services and applications. As users might not know upfront the number and details of the computations, they wish to offload their data once to the service provider and be contacted only to authorize the revelation of the result. Thus, in this work we present a solution that enables flexible computations on third-party signed data offloaded to a service provider in a privacy and integrity preserving manner.

To illustrate the inherent trade-off between privacy, integrity, and utility, we detail some of the use cases:

Smart Metering. Smart meters (i.e., data sources) measure the consumption of a user's household. The data is shared with a service provider (e.g., a different legal entity) for billing and load-balancing analysis. A user's privacy can be jeopardized as energy consumption patterns can reveal her habits [27, 68]. The service provider wants guarantees on the data integrity to provide reliable services [10]. *Malicious* users might cheat to reduce their bills or disrupt the service provider's analysis.

Disease Susceptibility. Medical centers and direct-to-consumer services [3, 92], provide a user with her DNA sequence to improve her health and to customize her treatments. Genomic data can be used for disease-susceptibility tests offered by service providers, e.g., research institutions that seek to form the appropriate cohorts for their studies. The user wants to protect her data as DNA is considered a very sensitive and immutable piece of information for her and her relatives [45]. Correspondingly, service providers are keen on collecting users' data and verifying its integrity so that they can use it for disease-risk estimation or other types of analyses, e.g., drug-effect prediction or health certificates. Malicious users might tamper with the genomic data they share to disrupt this process and pass a medical examination.

Location-Based Activity Tracking. A user's wearable device monitors her location by querying location providers. The user then shares this information with service providers, e.g., online fitness social networks [63] or insurance companies [2] to obtain activity certificates or discount coupons. As location data can reveal sensitive information, e.g., her home/work places or habits [61, 93], the user is concerned about her privacy. Service providers want legitimate data to issue activity certificates, provide discounts for performance achievements, and build realistic user profiles. Malicious users might be tempted to modify their data, aiming to claim fake accomplishments and obtain benefits they are not entitled to.

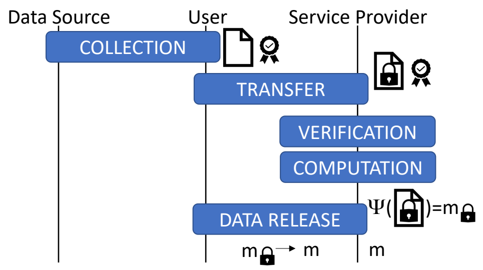


Figure 1: Three-party model and their interaction phases. 🗋 is the private information authenticated with 🏅. The user protects it via 🔒. The service provider computes $\psi(\cdot)$ on the protected data and obtains an output which is revealed as m.

The above use cases fall under the three-party model of Figure 1, with (i) malicious users, and (ii) honest-but-curious service providers and data sources; as such, they exhibit the trade-off between privacy, integrity, and utility. To support integrity protection regarding users' data, service providers require a data source to certify it, e.g., by means of a digital signature. This certification should require minimal to no changes to the data source: using only deployed hardware and software infrastructure. Another common denominator is that service providers want to collect users' data and perform various computations. Consequently, users should be able to offload their protected data to service providers (i.e., transfer a copy of the data only once) in such a way that their privacy is preserved, the data integrity can be verified, and various flexible computations are feasible.

A simple solution is to establish a direct communication channel between the data source and the service provider. This way, the data source could compute the operations queried by the service provider on the user's data. However, this would prevent the user from remaining in control of her data and require the data source to bear computations that are outside of its interests. Another approach is to let the data source certify the user's data by using specialized digital signature schemes such as homomorphic signatures [19, 24–26, 56] or homomorphic authenticators [5, 48, 54, 87]. Thus, the user could locally compute the queried operation and provide the service provider with the result and a homomorphic signature attesting its correct computation on her data. However, this would require software modifications at the data source, which would come at a prohibitive cost for existing services, and introduce significant overhead at the user.

In the existing literature, several works specialize in the challenges imposed by the above use cases but provide only partial solutions by either addressing privacy [12, 32, 40, 41, 71, 73], or integrity [20, 37, 83, 86]. The handful of works addressing both challenges require significant modifications to existing hardware or software infrastructures. For instance, SecureRun [85], which achieves privacy-preserving and cheat-proof activity summaries, requires heavy modifications to the network infrastructure. Similarly, smart metering solutions using secure aggregation, e.g., [4, 74, 76], rely on specialized signature schemes that are not yet widely supported by current smart meters. These approaches are tailored to their use case and cannot be easily adapted to others, hence there is the need for a *generic* solution to the trade-off between privacy and integrity, without significantly degrading utility.

ADSNARK [13] is a generic construction that could be employed to address the trade-off between privacy, integrity, and utility. In particular, it enables users to locally compute on data certified by data sources and to provide proof of correct computation to service providers. However, ADSNARK does not support the feature of data offloading that enables service providers to reuse the collected data and to perform various computations. Indeed, ADSNARK and other zero-knowledge solutions [17, 49, 50], require the user to compute a new proof every time the service provider needs the result of a new computation. Furthermore, it requires a trusted setup, and is not secure in the presence of quantum adversaries [66]. The latter should be taken into account considering recent advances in quantum computing [9] and the long term sensitivity of some data.

In this work we propose CRISP (privaCy and integRIty preServing comPutations), a novel solution that achieves utility, privacy, and integrity; it is generic, supports data offloading with minimal modification to existing infrastructures, relaxes the need for a trusted setup, and is quantum-resistant. Motivated by the need to protect users' privacy and by the offloading requirement to support multiple computations on their data, CRISP relies on quantum-resistant lattice-based approximate homomorphic encryption (HE) primitives [35] that support flexible polynomial computations on encrypted data without degrading utility. To ensure data integrity, we employ lattice-based commitments [15] and zero-knowledge proofs [29] based on the multi-party-computation-in-the-head (or MPC-in-the-head) paradigm [64], which enable users to simultaneously convince service providers about the correctness of the encrypted data, as well as the authenticity of the underlying plaintext data, using the deployed certification mechanism.

We evaluate our solution on three use cases covering a wide range of applications and computations: smart metering, disease susceptibility, and location-based activity-tracking. Our experimental results show that our construction introduces acceptable computation overhead for users to privately offload their data and for service providers to both verify its authenticity and to perform the desired computations. The magnitude of the communication overhead fluctuates between tens and hundreds of mega bytes per proof and is highly dependent on the use case and its security requirements. To this end, in Section 6, we also present different optimizations that can reduce the proof size, thus making our construction practical for real-life scenarios. Additionally, we demonstrate that CRISP achieves high accuracy in the computations required by the use cases, yielding an average absolute accuracy of more than 99.99% over the respective datasets. Compared to the state of the art [13], we reach comparable performance and achieve post-quantum security guarantees with more flexibility in the computations.

Our contributions are the following:

- A generic, quantum-resistant solution that enables privacy and integrity preserving computations in the three-party model of Figure 1, with minimal modifications of the existing infrastructure;
- the necessary primitives to achieve authenticity verification of homomorphically encrypted data in the quantum random oracle model;
- an implementation of CRISP [72] and its performance evaluation on various representative use cases that rely on different types of computations and real-world datasets.

To the best of our knowledge, it is the first time such a solution is proposed.

This paper is organized as follows: In Section 2, we discuss the system and threat model on which our construction operates. In Section 3, we introduce useful cryptographic primitives. Then, we present CRISP's architecture in Section 4 and in Section 5 we perform its privacy and security analysis. In Section 6, we evaluate CRISP on various use cases and in Section 7 we discuss some of its aspects. We review the related work in Section 8 and conclude in Section 9.

2 Model

We describe the model, assumptions, and objectives of CRISP.

2.1 System Model

We consider three entities: a user, a service provider, and a data source, as depicted in Figure 1. The user obtains from the data source certified data about herself and/or her activities, she subsequently shares it with the service provider to obtain some service. The user is interested in sharing (i.e., offloading) her data while protecting her privacy, i.e., she wants to have full control over it but still obtain utility from the service provider. The service provider is interested in (i) verifying the authenticity of the user's data, and (ii) performing on it multiple computations that are required to provide the service and/or improve its quality. The data source can tolerate only minimal changes to its operational process and cannot cope with any heavy modification to the underlying infrastructure and dependencies of the hardware and software. Finally, we assume the existence of a public key infrastructure that verifies the identities of the involved parties as well as secure communication channels between the user and the data source, and between the user and the service provider.

2.2 Threat Model

We present the assumed adversarial behavior for the three entities of our model with computationally bounded adversaries.
Data Source. The data source is considered honest and is trusted to generate valid authenticated data about the users' attributes or activities.
Service Provider. The service provider is considered *honest-but-curious*, i.e., it abides by the protocol and does not engage in denial-of-service attacks. However, it might try to infer as much information as possible from the user's data and perform computations on it without the user's consent.
User. We consider a *malicious but rational* user. In other words, she engages in the protocol and will try to cheat only if she believes that she will not get caught – and hence be identified and banned – by the service provider. This type of adversary is also referred to as *covert* in the literature [11]. The user is malicious in that she might try to modify her data, on input or output of the data exchange, in order to influence the outcome of the service provider's computations to her advantage. Nonetheless, the user is rational, as she desires to obtain utility from the service provider and thus engages in the protocol.

2.3 Objectives

Overall, the main objective of our construction is to provide the necessary building blocks for secure and flexible computations in the considered three-party model. To this end, user's privacy should be protected by keeping her in control of the data even in a post-quantum adversarial setting, and the service provider's utility should be retained by ensuring the integrity of the processed data. The above objectives should be achieved by limiting the impact on already deployed infrastructures, thus, by requiring only minimal changes to the data source's operational process. More formally, the desired properties are: (a) **Utility**: Both user and service provider are able to obtain the correct result of a public computation on the user's private data; (b) **Privacy**: The service provider does not learn anything more than the output of the computation on the user's private data; and (c) **Integrity**: The service provider is ensured that the computation is executed on non-corrupted data certified by the data source.

3 Preliminaries

We introduce the cryptographic primitives used in Section 4 to instantiate CRISP. In the remainder of this paper, let $a \leftarrow \chi$ denote that a is sampled from a distribution χ; a vector be denoted by a boldface letter, e.g., $\mathbf{x}$, with $\mathbf{x}[i]$ its i-th element and $\mathbf{x}^T$ its transpose. For a complex number $z \in \mathbb{C}$, we denote by $\bar{z}$ its conjugate. Moreover, let $\|$ denote the concatenation operation, $\boldsymbol{I}_n$ the identity matrix of size n, and $\mathbf{0}_k$ a vector of k zeros.

3.1 Approximate Homomorphic Encryption

Homomorphic encryption is a particular type of encryption that enables computations to be executed directly on ciphertexts. The most recent and practical homomorphic schemes rely on the hardness of the Ring Learning with Errors (RLWE) problem which states that, given a polynomial ring $\mathcal{R}_q$, for a secret polynomial s, it is computationally hard for an adversary to distinguish between $(a, a \cdot s + e)$ and (a, b), where e is a short polynomial sampled from a noise distribution, and a, b are polynomials uniformly sampled over $\mathcal{R}_q$.

Cheon *et al.* recently introduced the CKKS cryptosystem [35] (improved in [33]), an efficient and versatile leveled homomorphic scheme for approximate arithmetic operations. An approximate homomorphic encryption scheme enables the execution of approximate additions and multiplications on ciphertexts without requiring decryption. It uses an isomorphism between complex vectors and the plaintext space $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N+1)$, where q is a large modulus, and N is a power-of-two integer. The decryption of a ciphertext yields the input plaintext in $\mathcal{R}_q$ with a small error. This small error can be seen as an approximation in fixed-point arithmetic.

In CKKS, given a ring isomorphism between $\mathbb{C}^{N/2}$ and $\mathbb{R}[X]/(X^N+1)$, a complex vector $\boldsymbol{z} \in \mathbb{C}^{N/2}$ can be encoded into a polynomial m denoted by a vector $\mathbf{m}$ of its coefficients $\{m_0, \ldots, m_{N-1}\} \in \mathbb{R}^N$ as $\mathbf{m} = \frac{1}{N}(\bar{\boldsymbol{U}}^T \cdot \boldsymbol{z} + \boldsymbol{U}^T \cdot \bar{\boldsymbol{z}})$, where $\boldsymbol{U}$ denotes the $(N/2) \times N$ Vandermonde matrix generated by the $2N$-th root of unity $\zeta_j = e^{5^j \pi i/N}$. This transformation is extended to $\mathcal{R}_q$ by a quantization. Then, considering a maximum number of levels L, a ring modulus $q = \prod_{i=0}^{L-1} q_i$ is chosen with $\{q_i\}$ a set of number theoretic transform (NTT)-friendly primes such that $\forall i \in [0, L-1]$, $q_i = 1 \mod 2N$.

Let $\chi_{\text{err}}, \chi_{\text{enc}}$, and χ_{key}, be three sets of small distributions over $\mathcal{R}_q$. Then, for an encoded plaintext $m \in \mathcal{R}_q$, the scheme works as follows:

KeyGen(λ, N, L, q): for a security parameter λ and a number of levels L, generate $\boldsymbol{sk} = (1, s)$ with $s \leftarrow \chi_{\text{key}}$, $\boldsymbol{pk} = (b, a)$ with $a \leftarrow \mathcal{R}_q$, $b = -a \cdot s + e \mod q$, and $e \leftarrow \chi_{\text{err}}$. Additional keys which are useful for the homomorphic computations (i.e., rotation, evaluation keys, etc.) are denoted by $\boldsymbol{evk}$. We refer the reader to [59] for further details.

Encryption($m, \boldsymbol{pk}$): for $r_0 \leftarrow \chi_{\text{enc}}$ and $e_0, e_1 \leftarrow \chi_{\text{err}}$, output $\boldsymbol{ct} = (ct_0, ct_1) = r_0 \cdot \boldsymbol{pk} + (m + e_0, e_1) \mod q$.

Decryption($\boldsymbol{sk}, \boldsymbol{ct}$): Output $\hat{m} = \langle \boldsymbol{ct}, \boldsymbol{sk} \rangle \mod q_l$, where $\langle \cdot, \cdot \rangle$ denotes the canonical scalar product in $\mathcal{R}_{q_l}$ and l the current level of the ciphertext.

For brevity, we denote the above three operations as KeyGen(λ, N, q), Enc$_{\boldsymbol{pk}}(m)$, and Dec$_{\boldsymbol{sk}}(\boldsymbol{ct})$, respectively. The scheme's parameters are chosen according to the security level required (see [28]) to protect the inputs and *privacy*.

3.2 BDOP Commitment

Baum *et al.* [15] proposed the BDOP commitment scheme, that enables us to prove in zero-knowledge certain properties of the committed values to a verifier. Based on lattices, this scheme also builds on a polynomial ring $\mathcal{R}_q = \mathbb{Z}_q/(X^N+1)$, with the notable exception that q is a prime that satisfies $q = 2d+1 \mod 4d$, for some power-of-two d smaller than N.

BDOP is based on the hardness assumption of the module Short Integer Solution (SIS) and module Learning with Error (LWE) [70] to ensure its *binding* and *hiding* properties. We refer the reader to [15] for more details. For a secret message vector $\boldsymbol{m} \in \mathcal{R}_q^{l_c}$, and for a commitment with parameters (n, k), two public rectangular matrices $\boldsymbol{A}'_1$ and $\boldsymbol{A}'_2$, of size $n \times (k-n)$ and $l_c \times (k-n-l_c)$ respectively, are created by uniformly sampling their coefficients from $\mathcal{R}_q$. To commit the message $\boldsymbol{m}$, we sample $\boldsymbol{r}_c \leftarrow \mathcal{S}_\beta^k$, where $\mathcal{S}_\beta^k$ is the set of elements in $\mathcal{R}_q$ with l_∞-norm at most β and bounded degree, and compute

$$\text{BDOP}(\boldsymbol{m}, \boldsymbol{r}_c) = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \end{pmatrix} \cdot \boldsymbol{r}_c + \begin{pmatrix} \mathbf{0}_n \\ \boldsymbol{m} \end{pmatrix},$$

with $\boldsymbol{A}_1 = [\boldsymbol{I}_n \| \boldsymbol{A}'_1]$ and $\boldsymbol{A}_2 = [\mathbf{0}_{l_c \times n} \| \boldsymbol{I}_{l_c} \| \boldsymbol{A}'_2]$.

The BDOP commitment scheme can be used, with a Σ-protocol, to provide a *bound proof*: proof that a committed value is in a bounded range [14]. The main rationale behind this is to prove in zero-knowledge that the committed value plus a small value has a small norm. Given a commitment $\boldsymbol{c} = \text{BDOP}(\boldsymbol{m}, \boldsymbol{r}_c)$, the prover computes a commitment for a vector of small values $\boldsymbol{\mu}$ as $\boldsymbol{t} = \text{BDOP}(\boldsymbol{\mu}, \boldsymbol{\rho})$ and commits to this commitment in an auxiliary commitment $c_{\text{aux}} = \mathcal{C}_{\text{aux}}(\boldsymbol{t})$. The verifier selects a challenge $d \in \{0, 1\}$ and sends it to the prover who verifies its small norm and eventually opens c_{aux}. The prover also opens $\boldsymbol{t} + d \cdot \boldsymbol{c}$ to $\boldsymbol{z} = \boldsymbol{\mu} + d \cdot \boldsymbol{m}$ and $\boldsymbol{r}_z = \boldsymbol{\rho} + d \cdot \boldsymbol{r}_c$. Upon reception, the verifier checks that $\text{BDOP}(\boldsymbol{z}, \boldsymbol{r}_z) = \boldsymbol{t} + d \cdot \boldsymbol{c}$ and that the norms are small. The protocol is repeated to increase soundness and can be made non-interactive using the Fiat-Shamir heuristic.

3.3 Zero-Knowledge Circuit Evaluation

Zero-knowledge circuit evaluation (ZKCE) protocols enable a user to prove the knowledge of an input that yields a public output on an arithmetic or Boolean circuit that implements a specific public function [29, 55]. A circuit is defined as a series of gates connected by wires. Based on the multi-party computation (MPC) *in-the-head* approach from Ishai *et al.* [64], ZKCE techniques emulate players and create a decomposition of the circuit. The secret is shared among the emulated players, who evaluate the circuit in a MPC fashion and commit to their respective states. The prover then reveals the states of a subset of players depending on the verifier's challenge. By inspecting the revealed states, the verifier builds confidence in the prover's knowledge.

In particular, ZKB++ [29] is a Σ-protocol for languages of the type $\{y \mid \exists x \text{ s.t. } y = \Phi(x)\}$, where $\Phi(\cdot)$ is the representation of the circuit. With randomized runs, the verifier builds confidence in the prover's knowledge of the secret. The number of iterations is determined according to the desired soundness: For instance, to prove the knowledge of a message that yields a specific SHA-256 digest, a security level of 128-bits requires 219 iterations. The proof size is linked to the number of iterations but also to the number of gates that require non-local computations (e.g., AND for Boolean circuits, multiplication for arithmetic ones). Compared to earlier work, i.e., ZKBoo [55], ZKB++ reduces the proof size by not sending information that can be computed by the verifier. The security of ZKB++ is based on the quantum random oracle model. Overall, it achieves the following properties: (a) *2-privacy*, opening two out of the three players' views to the verifier reveals no information regarding the secret input, (b) *soundness*, a correct execution yields a valid witness with soundness error linked to the number of iterations, and (c) *completeness*, an honest execution of ZKB++ ensures a correct output.

4 Architecture

We now present our construction that enables computations on third-party certified data in a privacy and integrity preserving manner. It builds on (i) CKKS to encrypt the data and enable computations on it, and (ii) MPC-in-the-head and BDOP commitments to simultaneously verify a custom circuit that checks the integrity of the data and its correct encryption. Its workflow is decomposed into five phases: *collection*, *transfer*, *verification*, *computation*, and *release*. (1) In the collection phase, the user obtains data about herself or her activities from the data source, along with a certificate that vouches for its integrity and authenticity. (2) The user then encrypts the data, generates a proof for correct encryption of the certified data, and sends it with the ciphertexts to the service provider. (3) The service provider verifies the proof in the verification phase. Then, (4) it performs the desired computations on it, and (5) communicates with the user to obtain the corresponding result in the release phase.

4.1 Collection Phase

In this phase, the user (identified by her unique identifier *uid*) collects from the data source certified data about herself or her activities. The data source certifies each user's data point $\boldsymbol{x}$ using a digital signature $\sigma(\cdot)$ that relies on a cryptographic hash function $H(\cdot)$ to ensure integrity. We opt for SHA-256 as the hash function due to its widespread use as an accepted standard for hash functions [81]; our solution works with any signature scheme building on it. For example, Bernstein *et al.* [18] recently proposed a quantum-secure signature scheme employing SHA-256. In more detail, the data source generates a payload $msg=\{nonce, uid, \boldsymbol{x}\}$ and sends to the user a message M_0 defined by: $M_0=\{msg, \sigma(H(msg))\}$.

4.2 Transfer Phase

In this phase, the user protects her certified data points with the CKKS homomorphic encryption scheme (see Section 3.1) and generates a proof of correct protection. To this end, CRISP employs a ZKCE approach to simultaneously prove the integrity of the underlying data and its correct encryption, i.e., to convince a service provider that the noises used for encryption did not distort the plaintexts. In particular, the user evaluates a tailored circuit $\mathcal{C}$ (depicted in Figure 2) that (i) computes the encryption of the data with the CKKS scheme, (ii) generates BDOP commitments to the noises used for encryption, and (iii) produces the hash digests of the messages signed by the data source to verify their integrity. For ease of presentation, we describe the circuit that processes one data point $\boldsymbol{x}$. However, this can easily be extended to a vector $\boldsymbol{d}$ obtained from multiple data points $\{\boldsymbol{x}_i\}$. The circuit's structure is publicly known and its public parameters are the encryption public information $\boldsymbol{pk}, \boldsymbol{U}, N$, the matrices $\boldsymbol{A}_1, \boldsymbol{A}_2$ used in the BDOP commitment scheme and its parameter n, and additional information such as the user's identifier. The circuit's private inputs are the user's secret data point $\boldsymbol{x}$ and nonce, the encryption private parameters r_0, e_0, and e_1, and the private parameters of the BDOP commitment scheme $\boldsymbol{r}_c$. These inputs are arithmetically secret-shared among the three simulated players, according to the ZKB++ protocol. The outputs of the circuit are the ciphertext $\boldsymbol{ct}$, the commitment to the encryption noises $\boldsymbol{C}_{\text{bdop}}=\text{BDOP}((r_0, e_0, e_1)^T, \boldsymbol{r}_c)$, and the digest of the message $H(msg)$ signed by the data source.

CKKS and BDOP operate on slightly different polynomial rings, as described in Section 3. Consequently, we extend BDOP to the composite case where q is a product of NTT-friendly primes. We relax the strong condition on the challenge space from [15] that *all* small norm polynomials in $\mathcal{R}_q$ be invertible. This condition is required for additional zero knowledge proofs that are not used in our construction. We simply require that the challenge space of invertible elements be large enough to ensure the binding property of the commitment. In particular, considering that the divisors of zero in $\mathcal{R}_q$ are equally distributed in a ball $\mathcal{B}$ of norm β_c as in $\mathcal{R}_q$, the probability of having a non-invertible element when uniformly sampling from $\mathcal{B}$ is at most $\frac{N \cdot L}{2^l}$, where L is the number of prime factors in q, each having at least l bits. As a result, the number of invertible elements in $\mathcal{B}$ is lower-bounded by $|\mathcal{B}| * (1 - \frac{N \cdot L}{2^l})$, where $|\mathcal{B}|=(\beta_c+1)^N$ is the cardinality of the ball. Thus, by adequately choosing β_c and the product of primes, we create a sufficiently large challenge set of small-norm invertible elements in $\mathcal{R}_q$ (e.g., $> 2^{256}$). Moreover, we note that our circuit requires computations to be executed on the underlying arithmetic ring $\mathbb{Z}_q$ used for the lattice-based encryption and commitment schemes, as well as a Boolean ring $\mathbb{Z}_2$ for the computation of the SHA-256 hash digests. We also design a block that converts MPC-in-the-head arithmetic shares of the input data of the circuit into Boolean ones.

Overall, our circuit $\mathcal{C}$ consists of four blocks, showed in Figure 2: encryption, commitment, conversion, and hash block.

Encryption Block. This block operates in the arithmetic ring $\mathbb{Z}_q$ and takes as inputs the vector of integers in $\mathbb{Z}_q$ derived by quantization from the plaintext $\boldsymbol{x}$ produced during the data

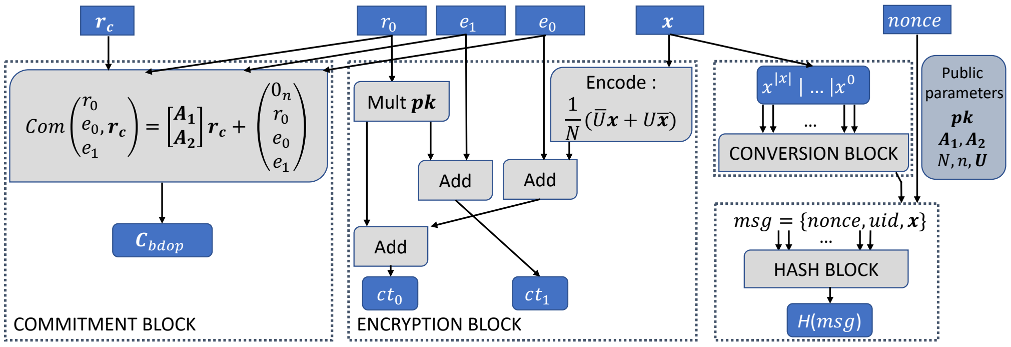

r_c
r_0
e_1
e_0
x
$nonce$
$Com\begin{pmatrix} r_0 \\ e_0, r_c \\ e_1 \end{pmatrix} = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} r_c + \begin{pmatrix} 0_n \\ r_0 \\ e_0 \\ e_1 \end{pmatrix}$
C_{bdop}
COMMITMENT BLOCK
Mult pk
Encode : $\frac{1}{N}(\bar{U}x + U\bar{x})$
Add
Add
Add
ct_0
ct_1
ENCRYPTION BLOCK
$x^{|x|}$ | ... | x^0
CONVERSION BLOCK
Public parameters pk A_1, A_2 N, n, U
$msg = \{nonce, uid, x\}$
HASH BLOCK
$H(msg)$

Figure 2: Overview of the verification circuit $\mathcal{C}$. Its inputs are denoted by rectangles and its outputs by rounded rectangles.

collection phase (see Section 4.1), as well as the encryption with private noise parameters r_0, e_0, and e_1. It first encodes the secret input data to a polynomial $m \in \mathcal{R}_q$ before computing the ciphertext $\boldsymbol{ct} = (ct_0, ct_1) = r_0 \cdot \boldsymbol{pk} + (m + e_0, e_1) \mod q$. This step requires only affine operations that can be computed locally for each simulated player of ZKB++ protocol. The encryption block is depicted in the middle part of Figure 2.

Commitment Block. This block also operates in the arithmetic ring $\mathbb{Z}_q$; its inputs are the private parameters of the encryption (i.e., r_0, e_0, and e_1) and commitment (i.e., $\boldsymbol{r_c}$) schemes. As the commitment scheme has the same external structure as the encryption one, this block operates equivalently and returns BDOP$((r_0, e_0, e_1)^T, \boldsymbol{r_c})$, requiring only local operations at each simulated player. An overview of the commitment block is shown in the leftmost part of Figure 2.

Conversion Block. This block enables us to interface two types of circuits that would otherwise be incompatible when following a ZKCE approach. The main idea is to transform an arithmetic secret sharing into a Boolean secret sharing in the context of MPC-in-the-head. Let $[x]_B$ denote the Boolean sharing of a value x and $[x]_A$ its arithmetic one. An arithmetic additive secret sharing in $\mathbb{Z}_q$ splits x into three sub-secrets x_0, x_1, and x_2 such that $x = x_0 + x_1 + x_2 \mod q$. Let x_i^k, be the k-th bit of the arithmetic sharing of the secret x for player i. A Boolean sharing $[x]_B$ cannot be directly translated from $[x]_A$ as the latter does not account for the carry when adding different bits. Considering that the modulus q can be represented by $|q|$ bits, the conversion block generates $|q|$ Boolean sub-secrets $[y]_B^j = \{y_0^j, y_1^j, y_2^j\}_B$, such that

$$\forall j \in [1, |q|] : x^j = \bigoplus_{i=0}^{2} y_i^j,$$

where $\oplus$ denotes the XOR operation (i.e., addition modulo 2), and x^j is the j-th bit of x. When designing such a block in the MPC-in-the-head context, we must make the circuit (2,3)-decomposable and ensure the 2-privacy property, i.e., revealing two out of the three players' views to the verifier should not leak any information about the input.

To reconstruct the secret in zero-knowledge and obtain a bitwise secret sharing, the procedure is as follows: For every bit, starting from the least significant one, the conversion block computes (i) the sum of the bits held by each player, plus the carry from the previous bits, and (ii) the carry of the bit. The computation of the carry requires interaction between the different players (i.e., making the operation a "multiplicative" one), hence we design a conversion block with a Boolean circuit that minimizes the amount of multiplicative gates.

More precisely, we design a bit-decomposition block for MPC-in-the-head building on Araki *et al.* 's optimized conversion [8] between a power-of-two arithmetic ring and a Boolean ring. Let Maj$(\cdot)$ be the function returning the majority bit among three elements. Then, the conversion circuit, for every bit $k \in [1, |x|]$, does the following:

1. locally reads $[\alpha_k]_B = \{x_0^k, x_1^k, x_2^k\}$ (i.e., for each player);
2. computes the first carry $[\beta_k]_B$ amongst those inputs:
$$\beta_k = \text{Maj}(x_0^k, x_1^k, x_2^k) = (x_0^k \oplus x_2^k \oplus 1)(x_1^k \oplus x_2^k) \oplus x_1^k;$$
3. computes the second carry $[\gamma_k]_B$ among those inputs with $\gamma_0 = \beta_0 = 0$:
$$\gamma_k = \text{Maj}(\alpha_k, \beta_{k-1}, \gamma_{k-1}) = (\alpha_k \oplus \gamma_{k-1} \oplus 1)(\beta_{k-1} \oplus \gamma_{k-1}) \oplus \beta_{k-1};$$
4. sets the new Boolean sharing of the secret to
$$[y]_B^k = [\alpha_k] \oplus [\beta_{k-1}] \oplus [\gamma_{k-1}].$$

To the best of our knowledge, this is the first time a bit-decomposition circuit is used for MPC-in-the-head, which enables to interface circuits working in different rings.

Hash Block. This block uses the SHA-256 circuit presented in [55] to compute the hash digest of the message $msg = \{nonce, uid, \boldsymbol{x}\}$ signed by the data source in the collection phase.

Full Circuit. With the above building blocks, and following the ZKB++ protocol, the user generates a proof that can convince the service provider that she has not tampered with the data obtained by the data source.

Furthermore, using BDOP's bound proof protocol (see Section 3.2) the user produces a proof of correct encryption, i.e., that the encryption noise has not distorted the underlying plaintext. The cryptographic material of the combined proofs (ZKCE & BDOP) is denoted by $\mathcal{P}$. At the end of the transfer phase, the user sends to the service provider the message: $M_1 = \{\boldsymbol{ct}, \boldsymbol{C}_{\text{bdop}}, \mathcal{P}, H(msg), \sigma(H(msg))\}$.

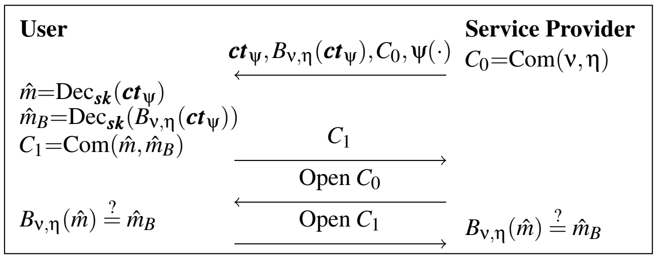

User
Service Provider
$\boldsymbol{ct}_\psi, B_{\nu,\eta}(\boldsymbol{ct}_\psi), C_0, \psi(\cdot)$
C_0=Com(ν,η)
$\hat{m}$=Dec$_{\boldsymbol{sk}}(\boldsymbol{ct}_\psi)$
$\hat{m}_B$=Dec$_{\boldsymbol{sk}}(B_{\nu,\eta}(\boldsymbol{ct}_\psi))$
C_1=Com$(\hat{m},\hat{m}_B)$
C_1
Open C_0
$B_{\nu,\eta}(\hat{m}) \overset{?}{=} \hat{m}_B$
Open C_1
$B_{\nu,\eta}(\hat{m}) \overset{?}{=} \hat{m}_B$

Figure 3: Release protocol for a computed value $\hat{m}$.

4.3 Verification Phase

Upon reception of a message M_1, the service provider verifies the signature using the provided hash digest. If satisfied, it verifies the proof $\mathcal{P}$ by first evaluating the circuit $\mathcal{C}$ following the ZKB++ protocol and then checking the bound proof for the encryption noises. Hence, it is assured that $\boldsymbol{ct}$ is the encryption of a data point $\boldsymbol{x}$ giving the hash that has been certified by the data source.

4.4 Computation Phase

Using the homomorphic capabilities of the CKKS encryption scheme, the service provider can perform any operation with a bounded predefined multiplicative depth (and arbitrary depth, with bootstrapping [34]) on validated ciphertexts received by the user. In particular, CKKS enables the computation of a wide range of operations on ciphertexts: additions, scalar operations, multiplications, and a rescaling procedure that reduces the scale of the plaintexts. Those functions enable the computation of polynomial functions on the ciphertexts. Moreover, it supports the evaluation of other functions such as exponential, inverse or square root [34–36], by employing polynomial approximations (e.g., least squares). Hence, the service provider can independently compute any number of operations on the user's encrypted data simply requiring interactions with the user to reveal their outputs (see Section 4.5).

4.5 Release Phase

At the end of the computation phase, the service provider holds a ciphertext of the desired output that can only be decrypted by the holder of the secret key. To this end, the service provider and the user engage in a two-round release protocol, which ensures the service provider that the decrypted output is the expected result of the computation on the user's data. The release protocol is depicted in Figure 3 and detailed next.

Let $\boldsymbol{ct}_\psi$ denote the ciphertext obtained by the service provider after performing computations on validated ciphertext(s), and $\hat{m}$ the corresponding plaintext. First, the service provider informs the user of the computation $\psi(\cdot)$ whose result it wants to obtain. Then, the service provider homomorphically blinds $\boldsymbol{ct}_\psi$ by applying the function $B_{\nu,\eta}(x)$=$\nu{\cdot}x$+η, with ν and η uniformly sampled in $\mathbb{Z}_q^*$ and $\mathbb{Z}_q$ resp., and commits to the secret parameters used for blinding (i.e., ν,η) using a hiding and binding cryptographic commitment Com$(\cdot)$ as C_0=Com(ν,η). A hash-based commitment scheme can be used for this purpose [29]. Subsequently, the service provider sends to the user the encrypted result $\boldsymbol{ct}_\psi$, its blinding $B_{\nu,\eta}(\boldsymbol{ct}_\psi)$, and the commitment C_0. Upon reception, the user checks if the function $\psi(\cdot)$ is admissible. If the user accepts the computation $\psi(\cdot)$, she decrypts both ciphertexts as: Dec$_{\boldsymbol{sk}}(\boldsymbol{ct}_\psi)$=$\hat{m}$ and Dec$_{\boldsymbol{sk}}(B_{\nu,\eta}(\boldsymbol{ct}_\psi))$=$\hat{m}_B$. Then, she commits to the decrypted results, i.e., C_1=Com$(\hat{m},\hat{m}_B)$, and communicates C_1 to the service provider who opens the commitment C_0 to the user (i.e., revealing ν,η). The user verifies that the initial blinding was correct by checking if $B_{\nu,\eta}(\hat{m})\overset{?}{=}\hat{m}_B$. If this is the case, she opens the commitment C_1 (i.e., revealing $\hat{m},\hat{m}_B$) to the service provider who verifies that the cleartext result matches the blinded information (i.e., by also checking if $B_{\nu,\eta}(\hat{m}) \overset{?}{=}\hat{m}_B$). At the end of the release phase, both parties are confident that the decrypted output is the expected result of the computation, while the service provider learns only the computation's result and nothing else about the user's data.

5 Privacy and Security Analysis

CRISP protects the user's privacy by revealing only the output of the agreed computation on her data, and it protects the service provider's integrity by preventing any cheating or forgery from the user. Here, we present these two properties and their corresponding proofs. The used lemmas and propositions are presented in Appendix A. A more detailed proof is available in the extended version of this paper [30].

5.1 Privacy

Proposition 5.1. *Consider a series of messages $\{msg_i\}$ certified by the data source with a digital signature scheme $\sigma(\cdot)$ that uses a cryptographic hash function $H(\cdot)$ with nonces. Assume that the parameters of the CKKS $(N,q,\chi_{enc},\chi_{key},\chi_{err})$ and BDOP (β,k,n,q,N) schemes have been configured to ensure post-quantum security, that the circuit $\mathcal{C}$ is a valid (2,3)-decomposition, and that the cryptographic commitment Com$(\cdot)$ is hiding and binding. Then, our solution achieves privacy by yielding nothing more than the result $\hat{m}$ of the computation on the user's data $\{\boldsymbol{x}_i\}$.*

Proof. To prove the privacy of CRISP, we construct an ideal simulator whose outputs are indistinguishable from the real outputs of CRISP's transfer and release phases.

Transfer Phase. In the quantum random oracle model (QROM), consider an ideal-world simulator $\mathcal{S}_t$ and any corrupted probabilistic polynomial time (PPT) service provider (i.e., the verifier). Without loss of generality, we consider only one round of communication between the user and service provider (i.e., one set of challenges). The simulator $\mathcal{S}_t$ generates a public-private key pair $(\boldsymbol{pk}',\boldsymbol{sk}')$. Following the encryption protocol, $\mathcal{S}_t$ samples $r'_0 \leftarrow \chi_{\text{enc}}$ and $e'_0, e'_1 \leftarrow \chi_{\text{err}}$ and computes the encryption of a random vector $\boldsymbol{m}'$ into $\boldsymbol{ct}'$. Similarly, it samples a commitment noise vector $\boldsymbol{r}'_c \leftarrow \mathcal{S}_\beta^k$ and commits (r'_0,e'_0,e'_1) into $\boldsymbol{C}'_{\text{bdop}}$. Using a random nonce, the simulator also hashes $H(\boldsymbol{m}'[0])$. Without loss of generality, this can be extended to all components of $\boldsymbol{m}'$. $\mathcal{S}_t$

then sends $\{\boldsymbol{ct}', \boldsymbol{C}'_{\text{bdop}}, H(\boldsymbol{m}'[0])\}$ to the service provider. The view of the service provider in the real protocol comprises $\{\boldsymbol{ct}, \boldsymbol{C}_{\text{bdop}}, H(msg)\}$. By the semantic security of the underlying encryption scheme [35], the hiding property of the BDOP commitment scheme (see Lemma A.2), and the indistinguishability property of the hash function in the QROM, the simulated view is indistinguishable from the real view.

Following the proof of Lemma A.3 in [14], for each iteration of the bound proof with challenge $\mathrm{d}\in\{0,1\}$, the simulator $\mathcal{S}_t$ can randomly draw z' and $\boldsymbol{r}'_z$ with small norm and set $\boldsymbol{t}=\text{BDOP}(z', \boldsymbol{r}'_z)-\mathrm{d}\boldsymbol{C}_{\text{bdop}}$ (see [14]). The simulator then commits to $\boldsymbol{t}$ in the bound proof protocol. Both ideal and real distributions are indistinguishable by the hiding property of the auxiliary commitment.

Following [55], and given a challenge $e \in \{1,2,3\}$, the simulator $\mathcal{S}_t$ uses the ZKB++ decomposition function on the inputs: $\boldsymbol{m}'$, e'_0, e'_1, r'_0 and $\boldsymbol{r}'_c$. It also samples random tapes $\boldsymbol{k}'_e$, $\boldsymbol{k}'_{e+1}$ used in the ZKB++ protocol. Then, $\mathcal{S}_t$ evaluates the arithmetic circuit according to: If gate c is linear, it defines view'^{c}_{e} and view'^{c}_{e+1} using the ZKB++ protocol (with view'^{c}_{e} the simulated state of gate c for player e). If gate c is a multiplication, it samples uniformly at random view'^{c}_{e+1} and computes view'^{c}_{e} using the ZKB++ protocol. Once the state of all the gates for players e and $e+1$ are defined, with respective outputs y'_e and y'_{e+1}, $\mathcal{S}_t$ computes $y'_{e+2} = y - (y'_e + y'_{e+1})$. Finally, the simulator returns the ZKB++ proof P' generated using the states of the simulated players, the random tapes $\boldsymbol{k}'_e$ and $\boldsymbol{k}'_{e+1}$, and the computed outputs y'_e, y'_{e+1}, and y'_{e+2}. The simulator $\mathcal{S}_t$ follows a protocol similar to the original ZKB++ protocol. The only difference is that for a multiplicative gate c, the simulated view value view'^{c}_{e+1} is sampled uniformly at random, whereas the original view value view^{c}_{e+1} is blinded by adding $R_i(c) - R_{i+1}(c)$, with $R_i(c)$ and $R_{i+1}(c)$ the outputs of a uniformly random function sampled using the tapes $\boldsymbol{k}_e$ and $\boldsymbol{k}_{e+1}$. Thus, the distribution of view^{c}_{e+1} is uniform and view'^{c}_{e+1} follows the same distribution in the simulation. Therefore, the ZKB++ simulator's output has the same distribution as the original transcript and the output of the simulator $\mathcal{S}_t$ is indistinguishable from the valid transcript to a corrupted verifier. Following the ideal functionality of $\mathcal{S}_t$, the ideal view of the service provider (i.e., $\{\boldsymbol{ct}', \boldsymbol{C}'_{\text{bdop}}, H(\boldsymbol{m}'[0]), P'\}$) is indistinguishable from the real view (i.e., $\{\boldsymbol{ct}, \boldsymbol{C}_{\text{bdop}}, H(msg), P\}$, with P the real ZKB++ proof). Thus, the ideal and real outputs are indistinguishable for the corrupted PPT service provider proving the privacy-property of CRISP's transfer phase. □

Release Phase. We construct a second simulator $\mathcal{S}_r$ to prove that CRISP's release protocol (Section 4.5) reveals nothing more than the result $\hat{m}$ to a curious verifier. A different simulator is required, as the release phase is independent from the transfer phase. We consider that $\mathcal{S}_r$ knows the blinding function ahead of time (i.e., it knows (ν,η)) for the real conversation leading to the service provider accepting $\hat{m}$. Upon reception of the first message $\{\boldsymbol{ct}_\psi, B_{\nu,\eta}(\boldsymbol{ct}_\psi), C_0, \psi(\cdot)\}$ such that $\text{Dec}_{\boldsymbol{sk}}(\boldsymbol{ct}_\psi) = \hat{m}$, $\mathcal{S}_r$ creates $\hat{m}_B$ using the blinding parameters. The simulator commits to $C'_1=\text{Com}(\hat{m}, \hat{m}_B)$, which is indistinguishable from C_1 to the curious verifier according to the hiding property of the commitment scheme. After receiving an opening for C_0, the simulator opens C'_1 to $\hat{m}$ and $\hat{m}_B$, which sustain the verifier checks as defined in Section 4.5. The binding property of the commitment scheme asserts that (ν,η) is used for the blinding. The aforementioned conversation between the prover and verifier is indistinguishable from the real conversation. By checking the function $\psi(\cdot)$, and as the service provider is honest-but-curious, the user is assured that the service provider evaluated $\psi(\cdot)$ and is not using her as a decryption oracle. If the user deems the function inadmissible, she aborts. □

5.2 Integrity

Proposition 5.2. *Consider a series of messages $\{msg_i\}$ certified by the data source with a digital signature scheme $\sigma(\cdot)$ that uses a cryptographic hash function $H(\cdot)$ with nonces. Assume that the parameters of the CKKS $(N, q, \chi_{enc}, \chi_{key}, \chi_{err})$ and BDOP (β, k, n, q, N) schemes have been configured to ensure post-quantum security, that the ZKB++ protocol execution of $\mathcal{C}$ achieves soundness κ, that the blinding function $B_{\nu,\eta}$ is hiding, and that the cryptographic commitment Com$(\cdot)$ is hiding and binding. Then, our solution achieves integrity as defined in Section 2.3, as it ensures with soundness κ that the output $\hat{m}$ is the result of the computation on the user's data.*

Proof. Let us consider a cheating user with post-quantum capabilities as defined in Section 2.2. She wants to cheat the service provider in obtaining from the public function $\psi(\cdot)$ a result that is not consistent with the certified data. The public function evaluated by the service provider is $\psi(\cdot)$ and returns $\hat{m}$ on the series $\{msg_i\}$ of data signed by the data source with the signature scheme $\sigma(\cdot)$. We interchangeably denote by $\psi(\cdot)$ the public function in the plaintext and ciphertext domains. By Lemma A.1, the ciphertext $\boldsymbol{ct}_\psi$ can be decrypted correctly using the secret key $\boldsymbol{sk}$. As stated in [55] adapted to [29], the binding property of the commitments used during the MPC-in-the-head guarantees that the proof $\mathcal{P}$ contains the information required to reconstruct the state of players e and $e+1$. Given three accepting transcripts (i.e., one for each challenge), the verifier can traverse the decomposition of the circuit from the outputs to the inputs, check every gate and reconstruct the input. By surjectivity of the ZKB++ decomposition function, the verifier can reconstruct x' s.t. $\Phi(x') = y$ proving the 3-special soundness property (see proof of Proposition A.1 in [55]). The completeness property of the ZKCE evaluation follows directly from the construction of the (2,3)-decomposition of the circuit. Thus, from a correct execution of τ iterations of the protocol (parameterized by the security parameter κ), a user attempting to cheat the ZKB++ execution will get caught by the service provider with probability at least $1-2^{-\kappa}$. Hence, a malicious but rational user can only cheat by tampering with the data before they are input to the circuit, i.e., the input messages or the encryption parameters. As the user is rational, she samples proper noise for the BDOP commitment; otherwise, she would lose either privacy or utility: not sampling noise uniformly from $\mathcal{S}_\beta^k$ would lead to a privacy leakage; conversely, sampling noises in $\mathcal{R}_q$ with norm bigger than β or with a degree above the threshold defined by the scheme would lead to an improperly formatted commitment, and thus, a potential loss in utility, as the service provider would reject it. By the collision-resistance property of the hash function, it is computationally infeasible for the user to find a collision and thus a tampered message yielding the same hash. By property A.3, the bound proof is correct and offers special soundness: the service provider will detect a cheating user that samples malicious noises to distort the encryption, with probability at least $1-2^{\kappa}$. Note that in the case of an abort, the protocol is simply re-executed. Finally, during the release protocol, the integrity of the computation's output $\hat{m}$ is protected by the hiding property of commitment C_0, the hiding property of the blinding function (seen as a one-time-pad shift cipher in $\mathbb{Z}_q$ which achieves perfect secrecy of $\nu{\cdot}x$, i.e., it is impossible for the user to find ν and blind another result as $\hat{m}_B$), and the binding property of C_1 [55]. Therefore, in CRISP users can only cheat with probability at most $2^{-\kappa}$. □

6 Evaluation

We evaluate CRISP on the three use cases discussed in Section 1, namely smart metering, disease susceptibility, and location-based activity tracking, using public real-world datasets. We detail the instantiation and parameterization of our proposed solution, then illustrate its overall performance per use case, in terms of both overhead and utility. As previously mentioned, CRISP enables to offload the data and to conduct multiple operations on it. For simplicity, we present only one operation per dataset.

6.1 Implementation Details

We detail how the various blocks of our construction are implemented and configured.

Implementation. We implement the various blocks of CRISP on top of different libraries. The homomorphic computations are implemented using the Lattigo library [43]. The commitment and encryption blocks of the circuit are implemented using CKKS from [89] by employing a ZKB++ approach. The circuit's Boolean part (i.e., the hash and conversion blocks) is implemented on top of the SHA-256 MPC-in-the-head circuit of [55]. All the experiments are executed on a modest Manjaro 4.19 virtual machine with an i5-8279U processor running at 2,4 GHz with 8 GB RAM.

CKKS & BDOP. For CKKS, we use a Gaussian noise distribution of standard deviation 3.2, ternary keys with i.i.d. coefficients in $\{0,\pm1\}^N$, and we choose q and N depending on the computation and precision required for each use case, such that the achieved bit security is always at least 128 bits. Each ciphertext encrypts a vector $\boldsymbol{d}$ consisting of the data points $\{x_i\}$ in the series of messages $\{msg_i\}$. Our three use cases need only computations over real numbers, hence we extend the real vector to a complex vector with null imaginary part. Similarly, the BDOP parameters for the commitment to the encryption noises are use case dependent. In principle, we choose the smallest parameters n and k to ensure a 128-bit security (n=1, k=5) and β is chosen according to N and q.

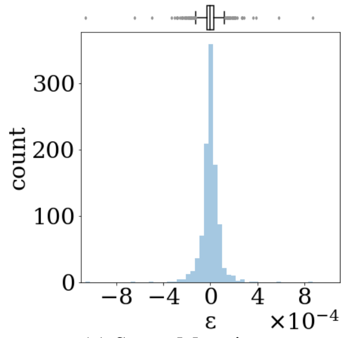
ZKB++. We set the security parameter κ to 128, which corresponds to 219 iterations of the ZKB++ protocol. We also consider seeds of size 128 bits and a commitment size of $|c|$=256 bits using SHA-256 as in [29]. Overall, considering the full circuit, the proof size per ZKB++ protocol iteration $|p_i|$ is calculated as

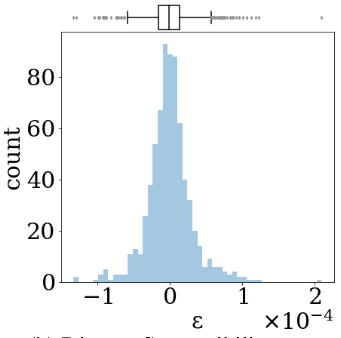
$$|p_i| = |c| + 2\kappa + \log_2 3 + \frac{2}{3}(|\boldsymbol{d}| + |\mathrm{Com}| + |\mathrm{Enc}| + |\boldsymbol{t}|) + b_{\mathrm{hash}} + b_{\mathrm{A2B}},$$

with $|\boldsymbol{d}|$ being the bit size of the secret inputs, $|\mathrm{Com}|$ the bit size of the commitment parameters, $|\mathrm{Enc}|$ the bit size of the encryption parameters, b_{hash} the number of multiplicative gates in the SHA-256 circuit, b_{A2B} the number of AND gates in the conversion block, and $|\boldsymbol{t}|$ the bit size of the additional information required to reconstruct the data source's message but not needed for the service provider's computation (e.g., user identifier, nonce, timestamps, etc.). We note that according to the NIST specification [81], SHA-256 operates by hashing data blocks of 447 bits. If the size of the user's input data exceeds this, it is split into chunks on which the SHA-256 digest is evaluated iteratively, taking as initial state the output of the previous chunk (see [81]). We adapt the SHA-256 Boolean circuit described in [55], which uses 22,272 multiplication gates per hash block, to the setting of ZKB++ [29]. The Boolean part of the circuit is focused on the $|\boldsymbol{x}|$ least significant bits of the arithmetic sharing of $\boldsymbol{d}$ which is concatenated locally with a Boolean secret sharing of the additional information (nonce, uid, etc.). In our implementation, the user needs 182 ms to run the Boolean part of the circuit associated with generating a hash from a 32-bits shared input $\boldsymbol{x}$. The verifier needs 73 ms to verify this part of the circuit.

Release Protocol. We use SHA-256 as a commitment scheme $\mathrm{Com}(\cdot)$ and a linear blinding operation $B_{\nu,\eta}(\cdot)$ in $\mathbb{Z}_q$.

Evaluation Metrics. We evaluate the performance of our solution on different use cases with varying complexity in terms of computation (i.e., execution time) and communication (i.e., proof size) overhead. The proof $\mathcal{P}$ is detailed as the proof for the ZKCE, as well as the BDOP bound proof. We also report the optimal ZKCE proof size per datapoint: i.e., if the ciphertexts are fully packed. To cover a wide range of applications we evaluate various types of operations on the protected data such as additions, weighted sums, as well as a polynomial approximation of the non-linear Euclidean distance computation. As CKKS enables approximate arithmetic, we measure the accuracy of our solution by using the relative error. Given the true output of a computation m and the (approximate)


(a) Smart Metering (Addition)


(b) Disease Susceptibility (Weighted Sum)

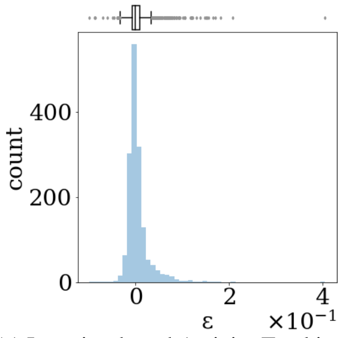

(c) Location-based Activity Tracking (Euclidean Distance)

Figure 4: Histogram and boxplot of the relative error for the three use cases. The boxes shown on top of each figure represent the interquartile range (IQ) and the median, the whiskers are quartiles $\pm 1.5 \cdot \text{IQ}$, and the dots are outliers.

value $\hat{m}$ computed with CRISP, the relative error ε is defined as $\varepsilon = \frac{m - \hat{m}}{m}$.

6.2 Smart Metering

We consider a smart meter that monitors the household's electricity consumption and signs data points containing a fresh nonce, the household identifier, the timestamp, and its consumption. The energy authority is interested in estimating the total household consumption (i.e., the sum over the consumption data points) over a specified time period I (e.g., a month or a year) for billing purposes

$$m_{\text{sm}} = \sum_{i \in I} \boldsymbol{d}[i],$$

where $\boldsymbol{d}$ is the vector of the household consumption per half hour. As our solution offloads the encrypted data to the service provider, additional computations, e.g., statistics about the household's consumption, are possible without requiring a new proof; this improves flexibility for the service provider.

Dataset & Experiment Setup. We use the publicly available and pre-processed UKPN dataset [90] that contains the per half hour (phh) consumption of thousands of households in London between November 2011 and February 2014. Each entry in the dataset comprises a household identifier, a timestamp, and its consumption phh. For our experiment, we randomly sample a subset of 1,035 households and estimate their total energy consumption over the time span of the dataset with our solution. We set the parameters as follows: We use a modulus of $\log q$=45 bits and a precision of 25 bits, which imposes a maximum of 2^{10} slots for the input vectors ($\log N$=11). Hence, each household's consumption phh is encoded with multiple vectors $\boldsymbol{d}_k$ to cover the time span of the dataset. To evaluate its proof size, we assume that the messages obtained from the smart meter include a 16-bit household id, a 128-bit nonce, a 32-bit timestamp, and a 16-bit consumption entry.

Results. The average time for encryption of a vector of 1,024 datapoints at the user side is t_{enc}=70 ms, and the decryption requires t_{dec}=0.7 ms. The mean time for the energy computation at the service provider side is t_{comp}=130 ms. To generate the proof for one ciphertext, containing 1,024 phh measurements (i.e., 21 days worth of data), the user requires t_{prove}=3.3 min, and its verification at the service provider's side is executed in t_{ver}=1.4 min. The estimated ZKCE proof size for each ciphertext of 1,024 elements is 643.4 MB, whereas the bound proof is 7.05 MB. For fully packed ciphertexts (1,024 datapoints), CRISP's proof generation and verification respectively take 195 ms and 80 ms per datapoint, with a communication of 628 KB. Finally, Figure 4a displays the accuracy results for the smart metering use case. We observe that our solution achieves an average absolute relative error of $5.1 \cdot 10^{-5}$ with a standard deviation of $7.2 \cdot 10^{-5}$, i.e., it provides very good accuracy for energy consumption computations. We remark that more than 75% of the households have an error less than $\pm 2.5 \cdot 10^{-4}$.

6.3 Disease Susceptibility

We assume a medical center that sequences a patient's genome and certifies batches of single nucleotide polymorphisms (SNPs) that are associated with a particular disease ∂. A privacy conscious direct-to-consumer service is interested in estimating the user's susceptibility to that disease by calculating the following normalized weighted sum

$$m_{\partial} = \sum_{i \in S_{\partial}} \omega_i \cdot \boldsymbol{d}[i],$$

where S_{∂} is the set of SNPs associated with ∂ and ω_i are their corresponding weights. The vector $\boldsymbol{d}$ comprises of values in $\{0, 1, 2\}$ indicating the presence of a SNP in 0, 1, or both chromosomes, which can be represented by two bits. This use case illustrates the need for flexibility in the service provider's computations, since it may be required to evaluate several diseases on the same input data at different times. Moreover, it accentuates the need for resistance against quantum adversaries, since genomic data is both immutable and highly sensitive over generations.

Dataset & Experiment Setup. We employ the 1,000 Genomes public dataset [1], that contains the genomic sequences of a few thousands of individuals from various populations. We randomly sample 145 individuals and extract 869

Table 1: Evaluation summary of CRISP (reported timings are the averages over 50 runs ± their standard deviation).

Use Case	Computation	Mean Absolute Relative Error	t_{enc} (ms)	t_{comp} (ms)	t_{dec} (ms)	Proof Size (MB)	t_{prove} (s)	t_{ver} (s)
Smart Metering	Sum	$5.1 \cdot 10^{-5}$	70 ± 10	130 ± 30	0.7 ± 0.3	650.5	200 ± 10	82 ± 5
Disease Susceptibility	Weighted Sum	$2.2 \cdot 10^{-5}$	60 ± 10	22 ± 5	2.7 ± 0.8	53.9	26 ± 4	13 ± 2
Location-Based Activity Tracking	Euclidean Distance	$1.5 \cdot 10^{-2}$	980 ± 70	180 ± 30	7 ± 2	1,603	470 ± 40	210 ± 10

SNPs related to five diseases: Alzheimer's, bipolar disorder, breast cancer, type-2 diabetes, and Schizophrenia. We obtain the weight of a SNP with respect to those diseases from the GWAS Catalog [21]. Then, for every individual, we estimate their susceptibility to each disease. For this use case, we use a precision $\log p{=}25$, a modulus of $\log q{=}56$ consumed over two levels and a polynomial degree of $\log N{=}12$. The input vector $\boldsymbol{d}$ (consisting of 2^{11} slots) is an ordered vector of integers containing the SNP values, coded on two bits, associated with the diseases. One vector is sufficient for the considered diseases. To estimate the proof size, we assume that the message signed by the data source contains a 16-bit user identifier, a 128-bit nonce, and the whole block of SNPs.

Results. The average encryption time for up to 2,048 SNPs at the user side is $t_{\mathrm{enc}}{=}60$ ms, and the decryption is $t_{\mathrm{dec}}{=}2.7$ ms. The computation time of the disease susceptibility at the service provider is $t_{\mathrm{comp}}{=}22$ ms. The user needs $t_{\mathrm{prove}}{=}26$ s to generate the proof for the arithmetic part of the circuit, and the service provider verifies it in $t_{\mathrm{ver}}{=}13$ s. The estimated proof size for the ZKCE is 36.6 MB, whereas the bound proof is 17.3 MB. Figure 4b shows our construction's accuracy for disease susceptibility computations by plotting the distribution of the relative error. We remark that the mean absolute relative error for such computations is appreciably low: $2.2{\cdot}10^{-5}$ on average with a standard deviation of $2.3{\cdot}10^{-5}$. Moreover, more that 75% of the evaluated records have an absolute error inside the range $\pm 0.7{\cdot}10^{-4}$.

6.4 Location-Based Activity Tracking

We assume that a user is running with a wearable device that retrieves her location points during the activity from a data source, e.g., a cellular network. The service provider, e.g., an online fitness social network, seeks to estimate the total distance that the user ran during her activity I:

$$m_{\mathrm{run}} = \sum_{i \in I} \sqrt{(\boldsymbol{d}[i+1]-\boldsymbol{d}[i])^2+(\boldsymbol{d}[\frac{N}{4}+i+1]-\boldsymbol{d}[\frac{N}{4}+i])^2},$$

with $\boldsymbol{d}$ the vector of UTM (Universal Transverse Mercator) inputs packing Eastings in the first half of the vector and Northings in the second. Given that Euclidean distance computations require the evaluation of a non-linear square root function, we consider its least-squares approximation by a degree seven polynomial on a Legendre polynomial base.

Dataset & Experiment Setup. We run our experiment on a public dataset from Garmin Connect [63]. This dataset contains GPS traces of thousands of users engaging in various activities such as walking, running, and cycling. We randomly sample 2,000 running traces and we discard traces with less than 15 points and more than 2,000 points. Our initial dataset analysis shows that the traces are very *noisy*: we identified unrealistic distances between consecutive points, timestamps and locations. We use GPSBabel [75], an open-source software, to interpolate the running traces such that the following criteria are met: (a) the maximum speed of a runner is less than 10 m/s, (b) the maximum distance between consecutive points is less than 30 m, and (c) the time delta between two points is less than 3 s, which are realistic for running activities. We remove traces whose time sampling was improperly executed by the data source (difference more than 10 s, standard deviation more than 5, or a zero inter-quartile at 75%), as well as traces with unacceptable idleness[1], and we convert the remaining GPS traces to UTM to obtain the Northings and Eastings geographic coordinates. Overall, we obtain a dataset of 1,608 traces (80% of the initial 2K running trace dataset) which on average contain 1,124 datapoints and we estimate their total distance with CRISP.

Considering the polynomial approximation required for the square root function, we set the size of the polynomial ring $N{=}2^{13}$ and a modulus $\log q{=}184$. To calculate the proof sizes, we assume that the messages obtained from the data source contain a 16-bit user identifier, a 128-bit nonce, a 32-bit timestamp, and 24-bit Easting/Northing coordinates.

Results. The encryption and decryption overhead for fully packed ciphertexts of up to 2,048 points at the user side is $t_{\mathrm{enc}}{=}980$ ms and $t_{\mathrm{dec}}{=}7$ ms, respectively, and the Euclidean distance computation at the service provider requires $t_{\mathrm{comp}}{=}180$ ms. For 2,048 datapoints, the user generates the proof for the arithmetic part of the circuit in $t_{\mathrm{prove}}{=}7.9$ min, and the service provider verifies it in $t_{\mathrm{ver}}{=}3.4$ min. Considering that each message signed by the data source is 96-bits, the proof size per trace for the ZKCE is 1,499.2 MB, and the bound proof is 103.7 MB. For our dataset, the average proof size is 922.1 MB considering the mean number of points in the traces. In Section 6.5, we will show how to reduce this proof size. With fully packed ciphertexts, CRISP's proof generation requires 230 ms per datapoint and 100 ms for its verification, at a communication cost of 732 KB. Finally, Figure 4c plots the relative error that we achieve for Euclidean distance computations. In particular, the average absolute relative error is $1.5{\cdot}10^{-2}$ with a standard deviation of $2.3{\cdot}10^{-2}$.

[1]Idleness of a trace is a situation where the interquartile at 25% of the instant speed is less than 0.3 m/s and the covered distance is less than 15 m.

In Figure 4c, we see that more than 75% of the evaluated traces have an absolute error between ± 0.04. We observe that the polynomial approximation of the square-root introduces errors higher than the other use cases. An improved accuracy can be achieved by increasing the polynomial degree, but this would require to increase upfront the encryption parameters (N, L, q) introducing additional communication and computation overhead to CRISP. The wider spread of the relative error is due to the variance of the datapoints. Indeed, our analysis shows that gait, time sampling, and skewness of the speed distribution are among the factors that influence the overall relative error of the computations.

6.5 Reducing the Communication Overhead

Table 1 summarizes CRISP's overhead for three use cases: smart metering, disease susceptibility, and location-based activity tracking. We observe that it introduces acceptable computational overhead at the user and service provider sides and that it achieves average absolute relative error between $2.2 \cdot 10^{-5}$ and 0.015 for the desired computations. We remark however that our construction uses post-quantum secure lattice-based cryptographic primitives, such as encryption and commitment, and the MPC-in-the-head approach, to ensure the integrity of the user's data transfer. These come at the price of an increased communication (i.e., proof size). Therefore, we propose several improvements that one could employ to reduce this overhead and illustrate them in Figure 5 for the smart metering and location-based activity tracking use cases.

Random Integrity Checks (RIC). A first optimization is to reduce the number of data points whose integrity is checked by the service provider. This introduces a trade-off between CRISP's security level and its communication overhead. In particular, a service provider can decide to check only a subset of the input data hashes in the data verification phase, as we assume *malicious but rational* users (Section 2.2) who will not cheat if there is a significant probability of getting caught. This is achieved through a sigma-protocol, that can be made non-interactive with the Fiat-Shamir heuristic: The user sends the ciphertext that encrypts all the datapoints (this can be seen as a commitment). Then, the service provider challenges a subset of datapoints to be hashed in the verification circuit. Such a strategy enables a service provider to tune the solution depending on the level of confidence it has in the user. In Figure 5 we observe how the proof size decreases as the service provider checks fewer data blocks. For instance, if the service provider checks 20% of the data blocks in the verification phase (RIC-20%), the proof size for location-based activity tracking drops from 1,499.2 MB to 497 MB (i.e., 243 KB/datapoint), whereas for smart metering it decreases from 643.4 MB to 142.2 MB (i.e., 139 KB/datapoint). This yields a reduction of more than 66% in the total ZKCE communication overhead. Computation times to generate and verify the proofs are also more than halved.

Batching (BG). Another improvement is to modify the way

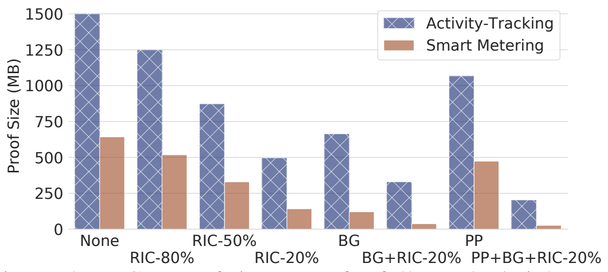


Figure 5: ZKCE proof size (MB) for fully packed ciphertexts and various optimizations.

data sources certify the users' data points. So far, in the smart metering and location-based activity tracking use cases, we assumed that data sources hash and sign every data point generated by the user. However, another strategy is to hash batches of data points in a single signed message. This modification is purely operational as it does not require additional software or hardware deployment. We set the batch size depending on the use case – i.e., considering the additional information of each message before signature – such that the overall batch can fit on a single SHA-256 input block of 447 bits. Figure 5 shows a reduction of more than 50% in proof size for the two use cases when batching (BG) is employed compared to the non-optimized solution. Batching can also be combined with RIC-20% (BG+RIC-20% in Figure 5): For smart metering, the ZKCE proof size is further reduced to 38.1 MB (i.e., 37.2 KB/datapoint), whereas for the location-based activity tracking the proof size drops to 329.7 MB (i.e., 161 KB/datapoint). For activity-tracking, the t_{prove} is reduced to 2.1 min and t_{ver} to 1.1 min (61 and 32 ms per datapoint, resp.). For smart metering, t_{prove} is reduced to 20 s and t_{ver} to 9.3 s (20 and 9 ms per datapoint, resp.).

ZKCE Pre-processing (PP). Finally, one can employ a ZKCE pre-processing model, such as that presented by Katz *et al.* [66]. The pre-processing model considers that the user executes *offline* a series of circuit evaluations on committed values. The service provider challenges a subset $\mathcal{M}$ of those evaluations and checks their integrity, and the remaining τ ones are used in an *online* phase along with the committed values. The rest of the protocol is similar to ZKB++. The proof size per iteration is reduced to:

$$|p_i| = 2\kappa + \tau \log_2 \frac{\mathcal{M}}{\tau} 3\kappa + \tau(\kappa \log_2 3 + 2\kappa + (|\boldsymbol{d}| + |\text{Com}| + |\text{Enc}| + |\boldsymbol{t}|) + 2(b_{\text{hash}} + b_{\text{A2B}})).$$

Regarding our three players setting, a 128-bit security level requires $\mathcal{M}=300$ and $\tau=81$, yielding a significant reduction of 25% on the proof size (see [66] for the computation details) compared to the non-optimized approach. Pre-processing, batching, and RIC can also be applied together to obtain smaller proofs (see PP+BG+RIC-20% in Figure 5): For smart metering, the ZKCE proof is reduced to 26.8 MB. Similarly, for location-based activity tracking, the ZKCE proof becomes 203.0 MB. This yields optimal ZKCE proof size per datapoint of 26.2 KB and 99.1 KB for smart metering and activity-tracking, respectively. Finally, we remark that according to

Katz *et al.* [66], a trade-off between proof size and prover's computations could be achieved by increasing the number of players involved in the MPC-in-the-head protocol. However, such an improvement would require additional changes in CRISP, e.g., the conversion block that interfaces the arithmetic and Boolean parts of the circuit should be adapted for a larger number of players.

6.6 Comparison with ADSNARK

A fair comparison with [13] is not trivial to achieve, as our solution provides post-quantum security and overcomes the constraint of a trusted setup. Nonetheless, here we provide hints of their qualitative and quantitative differences. In particular, ADSNARK considers a smart metering use case that requires a non-linear cumulative function for the billing analysis over a month of data. We consider a similar non-linear pricing function evaluated by a degree-two polynomial, and we evaluate CRISP on the UKPN dataset for 400 households, with $N=12$ and $\log q=106$. The median accuracy of our solution is higher than 99%. In terms of proof size, our construction yields 889.2 MB (verifying all phh measurements for a month), whereas the overhead induced by ADSNARK is 71 MB. However, we remark that the latter requires a new proof to be generated and exchanged every time a different computation is needed. In our solution, this cost is incurred only once; any subsequent operations can be computed locally by the service provider on the verified data. Additionally, ADSNARK accounts for only a "*theoretical estimate*" of the complexity of the signature circuit (with only 1K multiplicative gates for signature verification) and, if we were to evaluate our solution with this circuit, the proof size would be only 104.2 MB. Thus, our analysis shows that our construction offers comparable results to the state-of-the-art and provides stronger security guarantees.

7 Discussion

In this section, we present some interesting considerations that could influence the deployment of our solution.

Signature Scheme. As discussed in Section 4.1, CRISP is agnostic of the digital signature and it is compatible with any scheme that uses the SHA-256 hash function. We employ SHA-256 as it is widely deployed in current infrastructures, adopted by various signature schemes (e.g., the recent post-quantum SPHINCS [18] or the standard ECDSA schemes), and it is a benchmark for the evaluation of ZKBoo [55] and ZKB++ [29]. This flexibility enables CRISP to be compliant with currently deployed signature schemes that might not be quantum resistant (e.g., ECDSA) at the cost of CRISP's post-quantum integrity property. Working with other hash functions (e.g., SHA-3 that is employed in [29]) is possible, with modifications to CRISP's circuit.

Integrity Attacks. CRISP copes with malicious users that might attempt to modify their data or the computed result to their benefit. However, some use cases require accounting for additional threats. For example, for smart metering, users might purposefully *fail* to report some data (i.e., misreport) to reduce their billing costs. Similarly, in location-based activity-tracking, users might re-use pieces of data certified by the data source to claim higher performance and increase their benefits (i.e., double report). Such attacks can be thwarted by system level decisions; e.g., data sources can generate data points at fixed time-intervals known to service providers. Message timestamps can be encrypted along with the data points, so that service providers can verify their properties (e.g., their order or their range). As those attacks are application specific, we consider them out of the scope of this work.

Security of the ZKCE. Dinur and Nadler [42] unveil a vulnerability of ZKCE systems, such as ZKB++, to multi-target attacks on the pseudo-random number generators. However, as stated by the authors, these attacks require a very large number of protocol executions (more than 2^{57}) and thus are impractical and out of scope for our construction. The authors also argue that the use of appropriate salting in the pseudo-random number generation renders the attack very hard to succeed.

Multiple Users. CRISP trivially allows the service provider to compute aggregate statistics on data from multiple users by interacting separately with each of them and combining the results. Nonetheless, such a functionality can also be achieved by incorporating CKKS extensions to multiple parties. For instance, the multi-key scheme by Chen *et al.* [31] allows computations on ciphertexts generated by multiple users with their own keys. Alternatively, in the multiparty scheme by Mouchet *et al.* [80] users generate a common public key for which the corresponding private key is secret shared among them. In both cases, the service provider computes on the users' ciphertexts and interacts with all of them to decrypt the result.

Usability. Even though CRISP introduces non-negligible communication and computation overhead, it remains acceptable for modern systems. The independent iterations of the ZKCE make the proof generation highly paralellizable and require much less memory than the full proof size (experimentally, as little as 2 GB of RAM). CRISP has also the advantage of being an offline system that only requires interaction in the release protocol: e.g., the transfer phase can be executed when the user is idle. Additionally, recent communication systems such as fiber optic internet or 5G offer high throughput links: With a 80 Mb/s link, the proof for three weeks worth of smart metering data would only require about a minute to be transferred. For the activity tracking, CRISP can be executed when the user plugs her wearable device to a computer and transfer the data while recharging it.

8 Related Work

Although homomorphic encryption is a solution receiving much traction to protect privacy in various fields, such as machine learning [57, 65, 88] and medical research [67, 95], it only addresses the tension between privacy and utility, and it does not account for the authenticity of the encrypted data nor the correctness of the computation of its encryption (i.e., integrity). Verifiable encryption (VE) enables us to efficiently prove properties on encrypted data. Although VE solutions have been widely explored in the general case, notably by Camenish *et al.* [22, 23], they are still under investigation for lattice-based cryptographic systems that provide post-quantum security. Lyubashevsky and Neven [78] propose a *one-shot* verifiable encryption for short solutions to linear relations, i.e., a single run of their protocol convinces a verifier that the plaintext satisfies the relation. Recent improvements, e.g., [15] and [46], expand lattice-based VE to non-linear polynomial relations. Although VE can be used for proofs of correct encryption, it does not address data authenticity, which is ensured by cryptographic techniques, such as hash functions, that are more complex than polynomial relations.

To this end, homomorphic signatures [19, 24, 26, 56] and homomorphic authenticators [5, 48, 54, 87] enable privacy-preserving computations on authenticated data. In particular, such schemes produce a signature of the plaintext result of homomorphic computations without deciphering. In this setting, a data owner provides a signature to some protected data and sends it to a server for processing. The server generates a new valid signature for the result of the homomorphic computation, which yields nothing more than the message it is signing. Some works, e.g., [5], improve this area with constructions offering homomorphic signatures that cope with low-degree polynomial operations. Homomorphic authenticators could be a solution to the problem under investigation, but they (a) require data sources to employ non-widely-supported homomorphic signature schemes, thus violating the minimal infrastructure modification requirement of our model, and (b) do not support data offloading at the service provider to amortize communication and storage costs.

Verifiable computation (VC) [47, 51, 69] typically applies to cases where a computationally *weak* user transfers her encrypted data to a cloud provider that computes on it, and its objective is to ensure the correctness and trustworthiness of the result. As such, VC protects only the integrity of the cloud computations and not the authenticity of the user's provided data. Such techniques are orthogonal to CRISP and could be employed to enable users to verify the computations performed by service providers, if the latter are considered malicious (and not honest-but-curious as in our Threat Model – see Section 2.2).

Other VC techniques, known as zero-knowledge arguments, enable us to prove general statements about user private inputs. Pinocchio [84] and subsequent works on succinct non-interactive arguments of knowledge (SNARKS), e.g., [38], build on Quadratic Arithmetic Programs [52] and bilinear maps to provide efficient proofs with small verification complexity. Backes *et al.* [13] extend SNARKS to the case of certified data (ADSNARK) and apply them to the three party model considered in this work. Similarly, ZQL [49] and $Z\emptyset$ [50] present languages and compilers for data certification, client side computation, and result verification. But those solutions require computations from the user every time a new query is performed, thus not supporting data offloading. Furthermore, as pointed out by Katz *et al.* [66], SNARKS suffer from one or both of the following problems: (a) they require a trusted set-up, and (b) they are insecure against quantum attacks (due to the use of bilinear maps [13, 38, 49, 50, 84]). Interestingly, this holds also for the recent work of Gennaro *et al.* [53]; they use LWE homomorphic encryption to achieve post-quantum security of their encodings, but still rely on trusted setups through common reference strings [39] and q-PKE [58] assumptions. This trusted setup constraint is addressed by Wahby *et al.* [94], but their solution relies on the discrete log problem that is *de facto* not post-quantum secure. Finally, Ben-Sasson *et al.* [17] propose STARKs which achieve transparent and scalable arguments of knowledge by relying only on the collision-resistant hash and Fiat-Shamir heuristic assumptions. Although STARK-like systems solve a similar problem to ours, they follow a different approach where the bulk of the work is executed at the user side and no data offloading is considered, hence, multiple data computations require the creation of multiple proofs.

A radically different approach to proving statements in zero-knowledge comprises solutions based on multi-party computation (MPC) such as ZKBoo [55], ZKB++ [29], Ligero [6], and KKW [66]. These solutions are built on top of the MPC-in-the-head paradigm introduced by Ishai *et al.* [64] and provide plausibly post-quantum secure mechanisms to prove the knowledge of an input to a public circuit that yields a specific output, due to a cut-and-choose approach over several runs. Our construction follows this approach to convince the service provider about the integrity of the user's data and its encryption. In a concurrent work [16], Baum and Nof also present the use of MPC-in-the-head to prove lattice-based assumptions. However, their construction is based on a different problem (SIS, Short Integer Solution) and, unlike CRISP, does not address the integrity check of the encrypted payload.

Another potential solution for enabling privacy-friendly and integrity preserving computations on authenticated data is to consider trusted hardware, such as Intel SGX [7, 62, 79], to process the data. The secure enclave could be positioned at the user side (returning a result certified by the enclave) or at the service provider side (decrypting ciphertexts and returning only the result of the computation). However, those solutions impose different trust assumptions and we consider them orthogonal to our work.

Several works are devoted to protecting privacy for smart metering (e.g., see surveys [44, 96]). However, only some of them, e.g., [4, 74], address also the concern of data integrity and authenticity, by relying on custom homomorphic signature schemes. The applicability of such solutions is limited as, according to their technical specifications [91], smart meters cope with standard digital signatures, e.g., ECDSA [82]. Similarly, a number of works, e.g., [12,41,95], employ homomorphic encryption to protect genomic privacy and to perform disease-susceptibility computations. Their model considers a medical unit that sequences the DNA of the user, who in turn protects it via homomorphic encryption before sending it for processing to a third-party. These solutions do not address the issue of data integrity or authenticity. Finally, several works are dedicated to both privacy and integrity in location-based activity tracking [60, 77, 85, 86, 97, 98]. They also are either peer-based [97, 98], infrastructure-based [77, 85], or hybrid [60, 86]. SecureRun [85] offers activity proofs for estimating the distance covered in a privacy and integrity preserving manner. Nevertheless, the system's accuracy relies on the density of access points, and it achieves at best a median accuracy of 78% (compared to 99.9% with CRISP on a similar dataset).

9 Conclusion

Data sharing among users and service providers in the digital era incurs a trade-off between privacy, integrity, and utility. In this paper, we have proposed a generic solution that protects the interests of both users and service providers. Building on state-of-the-art lattice-based homomorphic encryption and commitments, as well as zero-knowledge proofs, our construction enables users to offload their data to service providers in a post-quantum secure, privacy and integrity preserving manner, yet still enables flexible computations on it. We evaluated our solution on three different uses cases, showing its wide potential for adoption. As future work, we will explore extending CRISP to malicious service providers by combining secure computation techniques with differential privacy.

Acknowledgements

We would like to thank our shepherd Ian Goldberg and the anonymous reviewers for their helpful feedback. We are also grateful to Henry Corrigan-Gibbs, Wouter Lueks, and the members of the EPFL Laboratory for Data Security for their helpful comments and suggestions. This work was supported in part by the grant #2017-201 (DPPH) of the Swiss strategic focus area Personalized Health and Related Technologies (PHRT), and the grant C17-16 (SecureKG) of the Swiss Data Science Center.

References

[1] "1000 Genomes | A Deep Catalog of Human Genetic Variation," https://www.internationalgenome.org/.

[2] "Sanitas Active app | Sanitas health insurance," https://www.sanitas.com/en/private-customers/contact-help/customer-portal-and-apps/active-app.html.

[3] 23andMe, "DNA Genetic Testing & Analysis - 23andme AU, DE, FR & EU," https://www.23andme.com/en-int/.

[4] A. Abdallah and X. S. Shen, "A Lightweight Lattice-Based Homomorphic Privacy-Preserving Data Aggregation Scheme for Smart Grid," *IEEE Transactions on Smart Grid*, 2018.

[5] J. H. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, A. Shelat, and B. Waters, "Computing on authenticated data," *Journal of Cryptology*, 2015.

[6] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam, "Ligero: Lightweight sublinear arguments without a trusted setup," in *ACM SIGSAC Conference on Computer and Communications Security – CCS*, 2017.

[7] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for CPU based attestation and sealing," in *International Workshop on Hardware and Architectural Support for Security and Privacy*, 2013.

[8] T. Araki, A. Barak, J. Furukawa, M. Keller, Y. Lindell, K. Ohara, and H. Tsuchida, "Generalizing the SPDZ Compiler For Other Protocols," in *ACM SIGSAC Conference on Computer and Communications Security – CCS*, 2018.

[9] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, and *et al.* , "Quantum supremacy using a programmable superconducting processor," *Nature*, 2019.

[10] M. R. Asghar, G. Dán, D. Miorandi, and I. Chlamtac, "Smart meter data privacy: A survey," *IEEE Communications Surveys & Tutorials*, 2017.

[11] Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries," in *Theory of Cryptography Conference*, 2007.

[12] E. Ayday, J. L. Raisaro, J.-P. Hubaux, and J. Rougemont, "Protecting and evaluating genomic privacy in medical tests and personalized medicine," in *Workshop on Privacy in the Electronic Society - WPES*, 2013.

[13] M. Backes, M. Barbosa, D. Fiore, and R. M. Reischuk, "ADSNARK: Nearly Practical and Privacy-Preserving Proofs on Authenticated Data," in *IEEE Symposium on Security and Privacy*, 2015.

[14] C. Baum and I. Damgard, "Efficient Commitments and Zero-Knowledge Protocols from Ring-SIS with Applications to Lattice-based Threshold Cryptosystems," *Cryptology ePrint Report 2016/997*, 2016.

[15] C. Baum, I. Damgård, V. Lyubashevsky, S. Oechsner, and C. Peikert, "More Efficient Commitments from Structured Lattice Assumptions," in *SCN*. Springer, 2018.

[16] C. Baum and A. Nof, "Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography," in *IACR PKC*. Springer, 2020.

[17] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable Zero Knowledge with No Trusted Setup," in *CRYPTO*, 2019.

[18] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, "The SPHINCS+ signature framework," in *ACM SIGSAC Conference on Computer and Communications Security - CCS*, 2019.

[19] D. Boneh and D. M. Freeman, "Homomorphic signatures for polynomial functions," in *EUROCRYPT*, 2011.

[20] J. Buchmann, M. Geihs, K. Hamacher, S. Katzenbeisser, and S. Stammler, "Long-term integrity protection of genomic data," *EURASIP J. Inf. Secur.*, 2019.

[21] A. Buniello, J. MacArthur, M. Cerezo, L. Harris, J. Hayhurst, and *et al.* , "The NHGRI-EBI GWAS catalog of published genome-wide association studies, targeted arrays and summary statistics 2019," https://www.ebi.ac.uk/gwas/home.

[22] J. Camenisch and I. Damgård, "Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes," in *ASIACRYPT*, 2000.

[23] J. Camenisch and V. Shoup, "Practical Verifiable Encryption and Decryption of Discrete Logarithms," in *CRYPTO*, 2003.

[24] D. Catalano and D. Fiore, "Practical Homomorphic MACs for Arithmetic Circuits," in *EUROCRYPT*, 2013.

[25] D. Catalano, D. Fiore, R. Gennaro, and L. Nizzardo, "Generalizing homomorphic macs for arithmetic circuits," in *Public-Key Cryptography*. Springer, 2014.

[26] D. Catalano, D. Fiore, and L. Nizzardo, "On the security notions for homomorphic signatures," in *Applied Cryptography and Network Security*, 2018.

[27] A. Cavoukian, J. Polonetsky, and C. Wolf, "SmartPrivacy for the smart grid: embedding privacy into the design of electricity conservation," *Identity in the Information Society*, 2010.

[28] M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, J. Hoffstein, K. Lauter, S. Lokam, D. Moody, T. Morrison *et al.*, "Security of homomorphic encryption," 2017.

[29] 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 *ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[30] S. Chatel, A. Pyrgelis, J. R. Troncoso-Pastoriza, and J.-P. Hubaux, "Privacy and integrity preserving computations with CRISP," 2020, https://arxiv.org/abs/2007.04025.

[31] H. Chen, W. Dai, M. Kim, and Y. Song, "Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference," in *ACM SIGSAC Conference on Computer and Communications Security - CCS*, 2019.

[32] L. Chen, R. Lu, and Z. Cao, "PDAFT: A privacy-preserving data aggregation scheme with fault tolerance for smart grid communications," *Peer-to-Peer networking and applications*, 2015.

[33] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A full RNS variant of approximate homomorphic encryption," in *International Conference on Selected Areas in Cryptography – SAC*. Springer, 2018.

[34] ——, "Bootstrapping for approximate homomorphic encryption," in *EUROCRYPT*. Springer, 2018.

[35] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *ASIACRYPT*. Springer, 2017.

[36] J. H. Cheon, D. Kim, D. Kim, H. H. Lee, and K. Lee, "Numerical method for comparison on homomorphically encrypted numbers," Cryptology ePrint Archive, Report 2019/417, 2019, https://eprint.iacr.org/2019/417.

[37] J. T. Chiang, J. J. Haas, and Y.-C. Hu, "Secure and precise location verification using distance bounding and simultaneous multilateration," in *ACM conference on Wireless network security*, 2009.

[38] C. Costello, C. Fournet, J. Howell, M. Kohlweiss, B. Kreuter, M. Naehrig, B. Parno, and S. Zahur, "Geppetto: Versatile verifiable computation," in *IEEE Symposium on Security and Privacy*, 2015.

[39] I. Damgård, "Efficient concurrent zero-knowledge in the auxiliary string model," in *EUROCRYPT*, 2000.

[40] G. Danezis and E. De Cristofaro, "Fast and Private Genomic Testing for Disease Susceptibility," in *Workshop on Privacy in the Electronic Society - WPES*, 2014.

[41] E. De Cristofaro, S. Faber, and G. Tsudik, "Secure genomic testing with size- and position-hiding private substring matching," in *Workshop on privacy in the electronic society – WPES*, 2013.

[42] I. Dinur and N. Nadler, "Multi-target attacks on the picnic signature scheme and related protocols," in *EUROCRYPT*. Springer, 2019.

[43] EPFL-LDS, "Lattigo 1.1.0," Online: http://github.com/ldsec/lattigo, Oct. 2019.

[44] Z. Erkin, J. R. Troncoso-pastoriza, R. Lagendijk, and F. Perez-Gonzalez, "Privacy-preserving data aggregation in smart metering systems: an overview," *IEEE Signal Processing Magazine*, 2013.

[45] Y. Erlich and A. Narayanan, "Routes for breaching and protecting genetic privacy," *Nature Reviews Genetics*, 2014.

[46] M. F. Esgin, R. Steinfeld, J. K. Liu, and D. Liu, "Lattice-Based Zero-Knowledge Proofs: New Techniques for Shorter and Faster Constructions and Applications," in *CRYPTO*, 2019.

[47] D. Fiore, R. Gennaro, and V. Pastro, "Efficiently verifiable computation on encrypted data," in *ACM SIGSAC*

Conference on Computer and Communications Security – CCS, 2014.

[48] D. Fiore, A. Mitrokotsa, L. Nizzardo, and E. Pagnin, "Multi-key homomorphic authenticators," in *ASIACRYPT*, 2016.

[49] C. Fournet, M. Kohlweiss, G. Danezis, and Z. Luo, "ZQL: A compiler for privacy-preserving data processing," in *USENIX Security Symposium*, 2013.

[50] M. Fredrikson and B. Livshits, "Zø: An optimizing distributing zero-knowledge compiler," in *USENIX Security Symposium*, 2014.

[51] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers," in *CRYPTO*, 2010.

[52] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct nizks without PCPs," in *EUROCRYPT*. Springer, 2013.

[53] R. Gennaro, M. Minelli, A. Nitulescu, and M. Orrù, "Lattice-based zk-snarks from square span programs," in *ACM SIGSAC Conference on Computer and Communications Security – CCS*, 2018.

[54] R. Gennaro and D. Wichs, "Fully Homomorphic Message Authenticators," in *ASIACRYPT*, 2013.

[55] I. Giacomelli, J. Madsen, and C. Orlandi, "ZKBoo: Faster zero-knowledge for boolean circuits," in *USENIX Security Symposium*, 2016.

[56] S. Gorbunov, V. Vaikuntanathan, and D. Wichs, "Leveled fully homomorphic signatures from standard lattices," in *ACM symposium on Theory of computing*, 2015.

[57] T. Graepel, K. Lauter, and M. Naehrig, "ML Confidential: Machine learning on encrypted data," in *Information Security and Cryptology – ICISC*, 2012.

[58] J. Groth, "Short pairing-based non-interactive zero-knowledge arguments," in *ASIACRYPT*, 2010.

[59] K. Han and D. Ki, "Better bootstrapping for approximate homomorphic encryption," Cryptology ePrint Archive, Report 2019/688, 2019, https://eprint.iacr.org/2019/688.

[60] R. Hasan and R. C. Burns, "Where have you been? Secure Location Provenance for Mobile Devices," *CoRR*, 2011, http://arxiv.org/abs/1107.1821.

[61] A. Hern, "Fitness tracking app Strava gives away location of secret US army bases," *The Guardian*, 2018.

[62] M. Hoekstra, R. Lal, P. Pappachan, V. Phegade, and J. Del Cuvillo, "Using innovative instructions to create trustworthy software solutions." *HASP@ ISCA*, 2013.

[63] G. International, "Garmin Connect | Free Online Fitness Community," https://connect.garmin.com/.

[64] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Zero-Knowledge Proofs from Secure Multiparty Computation," *SIAM Journal on Computing*, Jan. 2009.

[65] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *USENIX Security Symposium*, 2018.

[66] J. Katz, V. Kolesnikov, and X. Wang, "Improved non-interactive zero knowledge with applications to post-quantum signatures," in *ACM SIGSAC Conference on Computer and Communications Security – CCS*, 2018.

[67] A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon, "Logistic regression model training based on the approximate homomorphic encryption," *BMC Medical Genomics*, 2018.

[68] P. Kumar, Y. Lin, G. Bai, A. Paverd, J. S. Dong, and A. Martin, "Smart grid metering networks: A survey on security, privacy and open research issues," *IEEE Communications Surveys & Tutorials*, 2019.

[69] J. Lai, R. H. Deng, H. Pang, and J. Weng, "Verifiable computation on outsourced encrypted data," in *Computer Security – ESORICS*, 2014.

[70] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Designs, Codes and Cryptography*, 2015.

[71] K. Lauter, A. López-Alt, and M. Naehrig, "Private computation on encrypted genomic data," in *LATINCRYPT*, 2014.

[72] LDS, "CRISP's Implementation," 2020, https://github.com/ldsec/CRISP.

[73] C. Li, R. Lu, H. Li, L. Chen, and J. Chen, "PDA: a privacy-preserving dual-functional aggregation scheme for smart grid communications," *Security and Communication Networks*, 2015.

[74] F. Li and B. Luo, "Preserving data integrity for smart grid data aggregation," in *IEEE International Conference on Smart Grid Communications*, 2012.

[75] R. Lipe, "GPSBabel: convert, upload, download data from GPS and Map programs v1.6," https://www.gpsbabel.org/index.html.

[76] R. Lu, X. Liang, X. Li, X. Lin, and X. Shen, "EPPA: An efficient and privacy-preserving aggregation scheme for secure smart grid communications," *IEEE Transactions on Parallel and Distributed Systems*, 2012.

[77] W. Luo and U. Hengartner, "VeriPlace: A Privacy-aware Location Proof Architecture," in *SIGSPATIAL*, 2010.

[78] V. Lyubashevsky and G. Neven, "One-Shot Verifiable Encryption from Lattices," in *EUROCRYPT*, 2017.

[79] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution." *HASP@ ISCA*, 2013.

[80] C. Mouchet, J. Troncoso-Pastoriza, and J. Hubaux, "Multiparty homomorphic encryption: From theory to practice," *IACR Cryptol. ePrint Arch.*, 2020.

[81] NIST, "FIPS 180-4, Secure Hash Standard (SHS)."

[82] ——, "Special publication 800-78-4: Cryptographic algorithms and key sizes for personal identity verification," 2015, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-78-4.pdf.

[83] A. Oskarsdottir, G. Masson, and P. Melsted, "BamHash: a checksum program for verifying the integrity of sequence data," *Bioinformatics*, 2015.

[84] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *IEEE Symposium on Security and Privacy*, 2013.

[85] A. Pham, K. Huguenin, I. Bilogrevic, I. Dacosta, and J.-P. Hubaux, "SecureRun: Cheat-Proof and Private Summaries for Location-Based Activities," *IEEE Transactions on Mobile Computing*, 2016.

[86] S. Saroiu and A. Wolman, "Enabling new mobile applications with location proofs," in *ACM Workshop on Mobile Computing Systems and Applications*, 2009.

[87] L. Schabhüser, D. Butin, and J. Buchmann, "Context hiding multi-key linearly homomorphic authenticators," in *CT-RSA*. Springer, 2019.

[88] H. Shafagh, A. Hithnawi, L. Burkhalter, P. Fischli, and S. Duquennoy, "Secure Sharing of Partially Homomorphic Encrypted IoT Data," in *ACM Conference on Embedded Network Sensor Systems*, 2017.

[89] SNUcrypto, "HEAAN," 2019, https://github.com/snucrypto/HEAAN.

[90] UKPN, "SmartMeter Energy Consumption Data in London Households - London Datastore," https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households.

[91] United Kingdom Department of Business Energy and Industrial Strategy, "2017/0350/UK - notification of smart metering technical specifications for data communications company (DCC) system release 2," 2018.

[92] D. I. US, "Accurate DNA Test For Diet, Fitness, Health & Wellness," https://www.dnafit.com/us/.

[93] J. Valentino-DeVries, N. Singer, M. Keller, and A. Krolik, "Your Apps Know Where You Were Last Night, and They're Not Keeping It Secret - The New York Times," https://www.nytimes.com/interactive/2018/12/10/business/location-data-privacy-apps.html.

[94] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish, "Doubly-efficient zksnarks without trusted setup," in *IEEE Symposium on Security and Privacy*, 2018.

[95] S. Wang, Y. Zhang, W. Dai, K. Lauter, M. Kim, Y. Tang, H. Xiong, and X. Jiang, "HEALER: homomorphic computation of exact logistic regression for secure rare disease variants analysis in gwas," *Bioinformatics*, 2015.

[96] W. Wang and Z. Lu, "Cyber security in the smart grid: Survey and challenges," *Computer networks*, 2013.

[97] X. Wang, A. Pande, J. Zhu, and P. Mohapatra, "STAMP: Enabling Privacy-Preserving Location Proofs for Mobile Users," *IEEE/ACM Trans. Netw.*, 2016.

[98] Z. Zhu and G. Cao, "APPLAUS: A Privacy-Preserving Location Proof Updating System for location-based services," in *IEEE INFOCOM*, 2011.

A Propositions and Lemmas

For the sake of completeness, and for the reader's convenience, we present here the lemmas and propositions used in Section 5 to support the privacy and security analysis of our solution.

Lemma A.1 (Lemma 1 in CKKS [35])**.** *The encryption noise is bounded by* $B_{clean}{=}8\sqrt{2}\sigma N{+}6\sigma\sqrt{N}{+}16\sigma\sqrt{hN}$. *If* $\boldsymbol{c}{\leftarrow}Enc_{\boldsymbol{pk}}(m)$ *and* $m{\leftarrow}Ecd(\boldsymbol{z},\Delta)$ *for some* $\boldsymbol{z}{\in}\mathbb{Z}[i]^{N/2}$ *and* $\Delta{>}N{+}2B_{clean}$*, then* $Dcd(Dec_{\boldsymbol{sk}}(\boldsymbol{c})){=}\boldsymbol{z}$.

Ecd(.,.) (resp. Dcd(.)) is the encoding (resp. decoding) function, Δ the scaling factor, and h the Hamming weight of $\boldsymbol{sk}$.

Lemma A.2 (Lemma 4 in BDOP [14])**.** *Assume the distribution* $\mathcal{D}$ *and that* R_q, m *are chosen such that: 1) the min-entropy of a vector drawn from* $\mathcal{D}$ *is at least* $(k+1)\log|R_q|{+}\kappa$*, where* κ *is a (statistical) security parameter, and 2) the class of functions* $\{f_a|a{\in}R_q^m\}$ *where* $f_a(r){=}a\cdot r$ *is universal when mapping the support of* $\mathcal{D}$ *to* R_q*. Then, the scheme is statistically hiding.*

Recall that $\mathcal{D}$ denotes the distribution of an honest prover's randomness for commitments.

Lemma A.3 (Lemma 10 in BDOP [14])**.** *The protocol* Π_{Bound} *has the following properties:*

- ***Correctness:*** *The verifier always accepts an honest prover when the protocol does not abort. The probability of abort is at most* $2/\gamma+2/\gamma_x$.
- ***Special soundness:*** *On input a commitment* $\boldsymbol{c}$ *and a pair of transcripts* $(\boldsymbol{c},d,(c_{aux},\boldsymbol{t},z,\boldsymbol{r}_z))$, $(\boldsymbol{c},d',(c'_{aux},\boldsymbol{t}',z',\boldsymbol{r}'_z))$ *where* $d{\neq}d'$*, we can extract either a witness for breaking the auxiliary commitment scheme, or a valid opening of* $\boldsymbol{c}$ *where the message* x *has norm at most* $\gamma_x N\beta_x$.
- ***Honest-verifier zero-knowledge:*** *Executions of protocol* Π_{Bound} *with an honest verifier can be simulated with statistically indistinguishable distribution.*

We remind that γ is a constant that regulates the abort probability, γ_x a constant piloting the norm of $\boldsymbol{z}$, and β_x an upper bound on the norm of possible x.

Lemma A.4 (Lemma 3 in BDOP [14])**.** *From a commitment* $\boldsymbol{c}$ *and correct openings* $\boldsymbol{r}$, f, $\boldsymbol{r}'$, f' *to two different message vectors* $\boldsymbol{x}$, $\boldsymbol{x}'$*, one can efficiently compute a solution* $\boldsymbol{s}$ *with* $\|\boldsymbol{s}\|_\infty \leq 2Nm\gamma\beta\gamma_D^2$ *to the Ring-SIS problem instance defined by the top row of* $\boldsymbol{A}_1$.

Proposition A.1 (Proposition 4.2 in ZKBoo [55])**.** *The ZKBoo protocol is a* Σ*-protocol for the relation* R_Φ *with 3-special soundness.*