Efficient Zero-Knowledge Arguments For Paillier Cryptosystem

Borui Gong
The Hong Kong Polytechnic University
borui.gong@connect.polyu.hk

Wang Fat Lau
The Hong Kong Polytechnic University
wf-franky.lau@connect.polyu.hk

Man Ho Au*
The Hong Kong Polytechnic University
mhaau@polyu.edu.hk

Rupeng Yang
University of Wollongong
orbbyrp@gmail.com

Haiyang Xue
The Hong Kong Polytechnic University
haiyangxc@gmail.com

Lichun Li
Ant Group
lichun.llc@antgroup.com

***Abstract*—We present an efficient zero-knowledge argument of knowledge system customized for the Paillier cryptosystem. Our system enjoys sublinear proof size, low verification cost, and acceptable proof generation effort, while also supporting batch proof generation/verification. Existing works specialized for Paillier cryptosystem feature linear proof size and verification time. Using existing sublinear argument systems for generic statements (e.g., zk-SNARK) results in unaffordable proof generation cost since it involves translating the relations to be proven into an inhibitive large Boolean or arithmetic circuit over a prime order field. Our system does not suffer from these limitations.**

The core of our argument systems is a constraint system defined over the ring of residue classes modulo a composite number, together with novel techniques tailored for arguing binary values in this setting. We then adapt the approach from Bootle et al. (EUROCRYPT 2016) to compile the constraint system into a sublinear argument system. Our constraint system is generic and can be used to express typical relations in Paillier cryptosystems including range proof, correctness proof, relationships between bits of plaintext, relationships of plaintexts among multiple ciphertexts, and more. Our argument supports batch proof generation and verification, with the amortized cost outperforming state-of-the-art protocol specialized for Paillier when the number of Paillier ciphertext is in the order of hundreds.

We report an end-to-end prototype and conduct comprehensive experiments across multiple scenarios. Scenario 1 is Paillier with packing. When we pack 25.6K bits into 400 ciphertexts, a proof that all these ciphertexts are correctly computed is 17 times smaller and is 3 times faster to verify compared with the naive implementation: using 25.6K OR-proofs without packing. Furthermore, we can prove additional statements almost for free, e.g., one can prove that the sum of a subset of the witness bits is less than a threshold t. Another scenario is range proof. To prove that each plaintext in 200 Paillier ciphertexts is of size 256 bits, our proof size is 10 times smaller than the state-of-the-art. Our analysis suggests that our system is asymptotically more efficient than existing protocols, and is highly suitable for scenarios involving a large number (more than 100) of Paillier ciphertexts, which is often the case for data analytics applications.

* *corresponding author*

1. Introduction

Additive homomorphic encryption (AHE) allows encrypted messages to be added without decryption or knowledge of the secret key. It is widely used in multiparty computation [1]–[3], federated learning [4], [5], private information retrieval [6], [7], oblivious transfer [8], and electronic voting [9], [10] for privacy guarantee. Since this characteristic facilitates data flow across mutually distrusted organizations, many privacy-preserving data aggregation schemes[1] [11]–[13], [13]–[20] have been proposed.

Among these aggregation applications, we consider a 2-party aggregation[2] between data providers $\mathcal{P}_1$ and $\mathcal{P}_2$, in which $\mathcal{P}_1$ holds a set of binary vectors, $\{\mathbf{b}_i\} = \{(b_{i,1}, b_{i,2}, \ldots, b_{i,F})\}$, while $\mathcal{P}_2$ holds a set of weights, $\{W_i\}$, for each entity i. $\mathcal{P}_1$ aims to analyze its data with the help of $\mathcal{P}_2$. Such scenario constitutes a basic building block within machine learning algorithms [17], network traffic statistics [18], recommendation systems [19], frequency estimation [21], [22], and attribute-weighted sum [23], etc.

We can treat $\mathcal{P}_1$ as a proxy possessing data collected from multiple users, organizations, or devices while treating $\mathcal{P}_2$ as a weight provider. We use $b_{i,j}$ to indicate a particular record of entity i towards unit j and W_i to denote "weight" for each entity, where $i \in [1, M]$ and $j \in [1, F]$. They aim to jointly and securely compute the weighted sum for each unit j as,

$$S_j = \sum_i b_{i,j} * W_i = b_{1,j} * W_1 + b_{2,j} * W_2 + \cdots b_{M,j} * W_M.$$

1. We refer to various aggregation scenarios here, including but not limited to, computing sum, mean, minimum or maximum value, and counting frequency among other advanced statistics in machine learning.

2. Note that our focus is on diverse analytical scenarios where latency is not critical. For example, in our voter analysis example, our objective is to conduct statistical aggregations, not real-time tallying for an entire election.

Here, we restrict $b_{i,j}$ to be Boolean and W_i to be an integer. If the i-th entity supports/owns the j-th unit, then $b_{i,j} = 1$; otherwise, $b_{i,j} = 0$. Using a binary vector to represent the possession of attributes is quite common. For example, it can indicate whether a user has a certain disease [17], a phone has installed an app [17], a person has visited a specific country [18], or the presence in a certain restaurant [14], browsing history, and so on.

1.1. AHE for Secure Data Analytics - An Example

We take voter analysis as an example. Suppose pollster company $\mathcal{P}_1$ (e.g., CNN or Fox News) collects voter preferences pre or post election through interviews. $\mathcal{P}_2$ holds voters' personal information such as age or salary. $\mathcal{P}_1$ aims to gain an indication of the average age or income group favoring each candidate with the help of $\mathcal{P}_2$, without revealing raw data to each other. In this example, each entity represents a voter while each unit represents a candidate. $b_{i,j} = 1$ denotes that the i-th voter prefers the j-th candidate. W_i denotes the age or salary of the i-th voter. $\mathcal{P}_1$ intends to compute S_j. This can be viewed as frequency estimation [21], [22], which generalizes the boolean predicate of attribute-weighted sum [23] where W_i is Boolean.

To preserve data privacy, the aforementioned voter analysis can be conducted securely by utilizing AHE. $\mathcal{P}_1$ generates the AHE key, encrypts $\{\mathbf{b}_i\}$, and sends the ciphertext to $\mathcal{P}_2$. $\mathcal{P}_2$ computes the weighted sum (in the ciphertext domain) and returns the result to $\mathcal{P}_1$. $\mathcal{P}_1$ decrypts the result to obtain the total age of voters for each candidate. Finally, $\mathcal{P}_1$ divides this sum by the number of voters who aim to support that candidate to calculate the average age.

The Paillier cryptosystem [24] is a prominent example of AHE has been standardized by ISO [25]. Since Paillier supports a very large message space, typically 2048-bit, packing is often used [26], [27] to reduce ciphertext expansion. In more detail, assume that $M \cdot max\{W_i\} \leq U$, where U is a chosen slot size. $\mathcal{P}_1$ packs all records associated with the same entity into one Paillier plaintext m_i such that $m_i := \sum_j 2^{U*(j-1)} b_{i,j}$. $\mathcal{P}_1$ encrypts each m_i into c_i and sends them to $\mathcal{P}_2$. $\mathcal{P}_2$ then computes $\bar{C} = \prod_i c_i^{W_i}$ and sends the result back. $\mathcal{P}_1$ decrypts $\bar{C}$ and obtains $\bar{m}$. It can parse $\bar{m}$ to obtain $S_j = \sum_i b_{i,j} \cdot W_i$.

In this paper, we call U the slot size, and the number of records packed into one ciphertext the number of slots.

1.2. The Need for Proofs of Well-formedness

While the above approach protects the privacy of $\mathcal{P}_1$'s data, it does not guarantee the privacy of $\mathcal{P}_2$'s data. Specifically, a malicious $\mathcal{P}_1^*$ can obtain the weight of a specific entity, say, the i^*-th entity, by biasing the message structure. $\mathcal{P}_1^*$ computes m_{i^*} as, $m_{i^*} = \sum_j 2^{U*(j-1)+U/2} b_{i^*,j}$. Consequently, the value of W_{i^*} appears in the higher part of S_j (provided that $U >> S_j$). In other words, the above approach only provides honest-but-curious security. To guard against this kind of attack, $\mathcal{P}_2$ should require $\mathcal{P}_1^*$ to prove that all ciphertexts are well-formed. We list three requirements regarding the well-formedness in our running example of voter analysis.

- *Packing with Binary Messages.* To support Paillier with packing, $\mathcal{P}_1$ needs to prove that each $b_{i,j}$ is binary and it should be located in the correct position.
- *Equality Proof for Sum of Records.* In a plurality-at-large election, there are multiple, say, t, seats to be elected. Each voter can vote for at most t candidates. This kind of electoral system is utilized for electing Senate nominees in Canada (Alberta) [28], Federal Senate in Brazil [29], Council of States in Switzerland (2019) [30], and the election committee in Hong Kong [31]. To prove that c_i correctly encrypts entity i's preference, $\mathcal{P}_1$ should prove that there are at most t 1's in each voter's vote m_i.
- *Range Proof for Sum of Units.* Even if the protocol is secure, a voter's weight may be leaked from the output of the analysis. For example, if only one voter supports candidate j, S_j reveals that voter's weight. Therefore, $\mathcal{P}_2$ requires a proof that there are more than T 1's in the records $\{b_{i,j}\}$ towards each candidate (i.e., unit) j.

The challenge here is that the proof should leak no information about $\mathcal{P}_1$'s data, otherwise, it will compromise its data privacy. Zero-knowledge proof/argument (ZKP) systems [32] enable a prover to convince a verifier of the truth of a statement without revealing any additional information, making them ideal tools to mitigate the above tension.

1.3. Limitation of Existing ZKPs for Paillier

Developing ZKPs for different relations among Paillier plaintexts is valuable. A rich body of work has focused on range proofs [33], [34], proving the plaintext is 0 [35], multiplication [10], a sequence of power relations [36], and so on. However, none of the existing works specifically address Paillier with packing. Furthermore, even without packing, proof size and verification time are linear in the number of entities. In the case of the voter analysis scenario (or others discussed in Subsection 1.6), $\mathcal{P}_1$ may transmit its collected data for analysis on a daily basis, as observed in [37], [38]. Each election poll or evaluation campaign produces at least hundreds to thousands of data pieces each day. Moreover, if $\mathcal{P}_1$ commences the analysis after the data collection, the message volume could reach millions, as noted in [39], [40]. Consequently, a proof whose size does not scale linearly with the number of plaintexts would be beneficial. It is also important to note that $\mathcal{P}_1$ often collaborates with multiple weight providers, making a proof that is non-interactive, small, and efficient in verification highly desirable.

One might consider utilizing existing zk-SNARKs. Significant progress has been made recently in constructing efficient ZKPs supporting statements expressed in arithmetic (over a prime field) or Boolean circuits [41]–[52]. However, directly applying these ZKPs to our problem results in proof of unacceptable efficiency. The main problem is that the cost of utilizing an arithmetic or Boolean circuit to represent Paillier encryption is huge. Specifically, Paillier Encryption involves modular exponentiations over N^2, and it is unclear how to represent this operation efficiently using Boolean or addition/multiplication gates over a prime field. Thus, representing the well-formedness of Paillier ciphertext (with packing) will lead to an impractical circuit size. For example, proving one plaintext-ciphertext pair is valid involves a circuit with 13335083 gates, even in the modest setting of $|N| = 1024$.[3] More discussions are given in Section 8.

Looking ahead, the corresponding relation can be represented by 1536 constraints (i.e., gates) using our constraint system. The number grows to around three thousand constraints when N is 2048-bit.

1.4. Our Approach

The source of inefficiency in using existing zk-SNARKs lies in representing statements related to Paillier encryption with an arithmetic circuit over a prime field. In this paper, we investigate a different approach, namely, representing the statement to be proven using an arithmetic circuit over the ring of residue classes modulo a composite number ($\mathbb{Z}_{N^2}$), which matches the ciphertext space of the Paillier cryptosystem. Modular arithmetic can then be expressed using a simple gate, significantly simplifying the representation of Paillier encryption. Consequently, we investigate how to adapt the existing ZKPs for arithmetic circuits over a prime field into our setting.

The main obstacle when working in $\mathbb{Z}_{N^2}$ instead of a prime field is that it is unclear how to prove a message is binary. In a prime field, $b * (b - 1) = 0$ implies that b is binary. However, in our setting, non-trivial roots exist because N^2 is not prime. To solve this problem, we develop an innovative approach: the prover additionally provides the sum of a random subset (of the verifier's choice) of the witness "bits". If this sum is small, the verifier is convinced that all witness "bits" are binary.[4] To offer zero-knowledge, the sum is not provided in the clear but is masked by a small "noise" value.

1.5. Our Contributions

We propose several zero-knowledge arguments of knowledge for various relations within the Paillier cryptosystem, including: (1) the well-formedness of multiple Paillier ciphertexts with packing of binary messages; (2) an extension that additionally proves the number of one's in each ciphertext is no larger than a certain threshold and the number of one's in each unit exceeds a certain threshold; and (3) range proof of multiple Paillier ciphertexts. Their corresponding argument systems, ZKAoK^*, ZKAoK^+ and ZKAoK', are presented in Sections 4 and 5 respectively. Our proof system features sublinear proof size and efficient verification time, while maintain an acceptable proof generation time. Specifically, we made the following contributions.

- We design a constraint system defined over $\mathbb{Z}_{N^2}$ to represent correct encryption of Paillier cryptosystem with plaintext satisfying various properties. We show how to compile the constraint system into an zero-knowledge argument of knowledge (ZKAoK), enabling a prover to convince a verifier the knowledge of witnesses satisfying the constraint system without revealing extra information.
- We design new techniques to prove that a witness is binary, even when the constraint system is defined over $\mathbb{Z}_{N^2}$ for an RSA modulus N. We believe that our new techniques can be used for other scenarios where using arithmetic constraints over composite order field is desirable.
- Based on the above, we give efficient ZKAoKs suitable for aggregation applications. We implement an end-to-end prototype and conduct a series of experiments to examine its practicability. In the voter analysis scenario, when proving packed Paillier, our proof size (in ZKAoK^*) is 27x smaller than using a standard OR-proof [53] for proving 51.2K bits when $|N| = 2048$. For proving 800 messages as 256-bit numbers, our proof size (in ZKAoK') is 27x smaller compared with state-of-the-art range proof. Since our system is asymptotically more efficient, the difference is even larger with more ciphertexts. Please refer to TABLE 1 for detailed comparisons.

1.6. More Application Scenarios

We further explore potential scenarios that could benefit from our approaches. Practical applications, including transaction evaluation [55], [56], advertisement targeting [57], [58], and social relationship analysis [59], [60], may find our approaches advantageous. For instance, in transaction evaluation, our method enables companies to identify target consumers' income groups in collaboration with banks, where W_i represents their salary. Additionally, entities like Google [61], [62], which provide advertising campaigns based on online ad targeting [63], [64], can use our approach to assess campaign effectiveness. Typically, these entities utilize binary vectors to represent various marketing strategies. Their effectiveness can be privately evaluated through our method by calculating the average conversion value, where W_i denotes the cost paid by each consumer.

Beyond these specific applications, our methodology offers potential advantages in scenarios requiring the use of correctly-structured binary vectors, a common occurrence in fields such as machine learning [65], [66], cryptographic schemes [67], [68], image and audio processing [69], [70], graph theorems, and so on. For instance, utilizing binary (or one-hot) vectors to encode labels is a common practice in classification [71], [72] or regression tasks [73], [74], which require that each vector can only contain a "1" and others

3. We are unable to generate the circuit for $|N| = 2048$ on our PC with Intel Core i9-12900K CPU and 160 GB of memory.

4. One needs to show that all non-trivial roots are large.

TABLE 1. Summarization and Comparison of Our Protocols with State-of-the-art Approaches in Different Scenarios with $|N| = 2048$-bit setting. *The inputs for* ZKAoK^* *are packed Paillier messages.* ZKAoK' *is for range proving 256-bit plaintexts.* $\mathcal{N}_p$ *and* $\mathcal{N}_b$ *represent the number of proved plaintexts and bits, respectively. A "–" denotes non-applicability. In the scenario proving many binary records, the amortized cost measures the cost per bit, while in a range proof, it measures the cost per message.*

Scenario	Protocol	$\mathcal{N}_p$	$\mathcal{N}_b$	Amo. Proof Size	Amo. Proof Time	Amo. Verification Time
multiple binary records	OR-proof [53]	1	1	>16384 bits	35.31 ms	18.04 ms
	ZKAoK^*	800	51.2K	**605.08 bits**	0.70 s	**3.79 ms**
		1M	64M	**18.85 bits**	0.31 s	**2.01 ms**
		10M	640M	**7.90 bits**	0.31 s	**1.97 ms**
multiple range proofs	[33], [54]	1	-	> 128 KB	0.24 s	199.32 ms
	ZKAoK'	800	-	**4.73 KB**	44.91 s	249.60 ms
		1M	-	**932.84 bits**	20.03 s	**121.08 ms**
		10M	-	**232.12 bits**	19.46 s	**118.48 ms**

should be "0". Similarly, in computing a special case of inner product [67], a "selection vector" is required, where only one coordinate is set to 1. We would like to emphasize that any constructions involving (structured) binary vectors can gain adaptive security through our approach.

1.7. Related Works

As we provide a ZKAoK protocol for Paillier, we now review and compare existing ZKPs customized for Paillier. Existing works can be mainly divided into two categories, one focuses on proving the validity of an RSA modulus (i.e., Paillier public key) while the other focuses on proving plaintext relations (including range). We use $\mathsf{pk} = N$ to denote a Paillier public key and $\mathsf{PL.Enc_{pk}}(m; r)$ to denote Paillier encryption of message m with randomness r.

Proving the Validity of A Paillier Public Key. [75] first proposed a ZKP on a number that is the product of two safe prime integers. It can be directly used for proving a valid Paillier public key. [76] also sketched the folklore method of proving the validity of an RSA modulus where it proves that $gcd(N, \phi(N)) = 1$. Although the standard Paillier public key is generated from two prime numbers, this statement still suffices to provide all properties of Paillier (e.g., additive homomorphism) through this requirement (see [33] Sec. 3.1). Besides, [36] also suggests combining methods in [77] and [78] for proving an RSA composite (we refer [36] for more detailed discussions). This category of proofs is orthogonal to ours. Looking ahead, in our protocol, $\mathcal{P}_2$ (verifier) should request $\mathcal{P}_1$ (prover) to provide a proof that: 1). the Paillier encryption key N is the product of two primes p and q and 2). p and q are sufficiently large. The first condition can be proved using the protocol introduced in Section 5.2 of [75]. The second can be achieved by proving the knowledge of a discrete logarithm that lies in a given range, utilizing techniques in Section 2.2 of [75], or [79].

Proving Paillier Plaintexts Relation. Another class mainly focuses on proving relations among Paillier plaintexts. [35] gives a construction on proving knowledge of an encrypted plaintext. For proving that the plaintext is 0, [10] gives constructions which is actually a proof of Nth power in Paillier. For proving multiplicative relations among Paillier plaintexts, [10] gives constructions on Π_{mul}. That is, we can prove that a message is the product of two other messages. For proving a more advanced relation, a sequence of powers, [36] constructs Π_{pow} based on Π_{mul}. In [33], they bridged two different worlds, Paillier encryption and Elliptic curve groups, and proposed a zero-knowledge proof for language $\mathcal{R}_{\mathsf{pl-ec}}$. That is, it can prove that the message in a given Paillier ciphertext is the discrete log of a given Elliptic curve point. Later, [34] gives constructions on Paillier and Pedersen commitments. It proves that the same value is used in encryption and commitment schemes. Besides, [33] and [34] also gave a range proof that is customized for Paillier, where [33] was adapted from the range proof in [54]. However, to achieve efficiency, both of them can only give range proofs with slack (i.e., inexact proofs).

For devising an exact range proof in Paillier, one may utilize a tight range proof, such as [49], [54], with some adaption. Since these range proofs do not work on Paillier, an integer commitment scheme [80] is needed as a bridge. In more detail, let $\mathsf{Enc}(x)$ be the Paillier encryption of x. Let CMT_I be an integer commitment scheme in which efficient range proofs exist. To prove that x lies in an exact range, the prover first produces commitments $c_1 := \mathsf{Enc}(x)$ and $c_2 := \mathsf{CMT}_I(x)$. He then engages the following protocols to prove i) Π_{c1}: c_1 is a Paillier encryption of x ii) Π_{c2}: the same value, x, is committed in c_2 iii) Π_{range}: the committed value in c_2 is in some range. Π_{range} can be achieved by invoking the existing range proofs on c_2. Although the range proof is efficient, auxiliary commitments may give a lower bound on the proof size.

1.8. Paper Organization

The rest of the paper is organized as follows. We give a technical overview in Section 2 and Preliminaries in Section 3. Then we present our main protocol in Section 4. We give our range proof protocol in Section 5. Security analysis is given in Section 6. We evaluate the performance in Section 7, followed by discussions in Section 8.

2. Technical Overview of Our Results

We provide a technical overview of our solution (main protocol) that proves correctness of packed Paillier encryption with multiple binary messages. We further show

TABLE 2. Existing Zero-knowledge Proofs and Our Protocols Customized for Paillier Cryptosystem.

Paper	Proved Relation	Corresponding Protocol
[75]	$\mathcal{R}_{\mathsf{composite}} = \{(N, (p, q)) : N = p \cdot q \wedge p, q \textit{ are primes}\}$	$\Pi_{\mathsf{composite}}$
[76]	$\mathcal{R}_{\mathsf{rsa}} = \{(N, \phi(N)) : gcd(N, \phi(N)) = 1\}$	Π_{rsa}
[10]	$\mathcal{R}_{\mathsf{zero}} = \{((c, N), r) : c = \mathsf{PL.Enc}_{\mathsf{pk}}(0; r)\}$	Π_{zero}
	$\mathcal{R}_{\mathsf{mul}} = \{(c_1, c_2, c_3, N), (m_1, r_1, m_2, r_2, r_3) : c_1 = \mathsf{PL.Enc}_{\mathsf{pk}}(m_1; r_1) \wedge c_2 = \mathsf{PL.Enc}_{\mathsf{pk}}(m_2; r_2) \wedge c_3 = \mathsf{PL.Enc}_{\mathsf{pk}}(m_1 \cdot m_2; r_3)\}$	Π_{mul}
[35]	$\mathcal{R}_{\mathsf{enc}} = \{(c, N), (m, r) : c = \mathsf{PL.Enc}_{\mathsf{pk}}(m; r)\}$	Π_{enc}
[36]	$\mathcal{R}_{\mathsf{pow}} = \{(c_1, c_2, \cdots, c_d, N), (m, r_1, \cdots, r_d) : \forall i \in [1, d], c_i = \mathsf{PL.Enc}_{\mathsf{pk}}(m^i; r_i)\}$	Π_{pow}
[33]	$\mathcal{R}_{\mathsf{pl.range}} = \{(c, N), (m, r) : c = \mathsf{PL.Enc}_{\mathsf{pk}}(m; r) \wedge m \in \mathbb{Z}_q\}$	$\Pi_{\mathsf{pl.range}}$
	$\mathcal{R}_{\mathsf{pl-ec}} = \{(c, N, \mathbb{G}_{ec}, Q, G_{\mathsf{ec}}, q), (m, r) : c = \mathsf{PL.Enc}_{\mathsf{pk}}(m; r) \wedge Q = m \cdot G_{\mathsf{ec}} \wedge m \in \mathbb{Z}_q\}$	$\Pi_{\mathsf{pl-ec}}$
[34]	$\mathcal{R}_{\mathsf{pl-ped}} = \{(c, N, c', g, h, N'), (m, r, \rho) : c = \mathsf{PL.Enc}_{\mathsf{pk}}(m; r) \wedge c' = g^m \cdot h^\rho \mod N'\}$	$\Pi_{\mathsf{pl-ped}}$
	$\mathcal{R}_{\mathsf{pl.range'}} = \{(c, N), (m, r) : c = \mathsf{PL.Enc}_{\mathsf{pk}}(m; r) \wedge x \in \mathbb{Z}_q\}$	$\Pi_{\mathsf{pl.range'}}$
ours	$\mathcal{R}^* = \{(\{c_i\}_{i\in[1,\mathcal{N}_p]}, N), (m_i, r_i, b^{(i)}_{32(s-1)})_{i\in[1,\mathcal{N}_p], s\in[1,64]} : \forall\ i \in [1, \mathcal{N}_p] \textit{ and } s \in [1, 64], c_i = \mathsf{PL.Enc}_{\mathsf{pk}}(m_i, r_i) \wedge$ $m_i = \sum_{s\in[1,64]} 2^{32(s-1)} \cdot b^{(i)}_{32(s-1)} \mod N \wedge\ b^{(i)}_{32(s-1)} \in \{0, 1\}\}$	ZKAoK^*
	$\mathcal{R}' = \{(\{c_i\}_{i\in[1,\mathcal{N}_p]}, N, \beta), (m_i, r_i, b^{(i)}_k)_{i\in[1,\mathcal{N}_p], k\in[0,\lvert\beta\rvert-1]} : \forall\ i \in [1, \mathcal{N}_p],\ c_i = (1 + N)^{m_i} \cdot r_i^N \mod N^2 \wedge m_i \leq \beta\}$	ZKAoK'

how we can prove correctness of polynomially many such ciphertexts.

Representing Arithmetic Circuit. Following the terminology of [46], the statements to be proved are represented as a list of equations known as constraints. For example, the following list of 5 constraints represents the circuit shown in Fig. 1. Note that all constraints are modulo N^2 unless otherwise indicated.

$$\begin{aligned} a_1 * b_1 &= c_1 \\ a_2 * b_2 &= c_2 \\ a_3 * b_3 &= c_3 \\ 3 * c_1 &= a_3 \\ c_1 + c_2 &= b_3 \end{aligned}$$

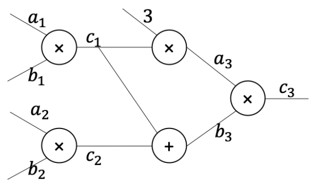

Figure 1. An Arithmetic Circuit

The knowledge of wire assignment to satisfy the circuit directly translates to the assignment of variables satisfying the set of constraints. There are two types of constraints, namely, multiplication and linear constraints. A multiplication constraint is of the form $a_x * b_x = c_x$ while a linear constraint is of the form $\sum_x w_x^{(a)} a_x + \sum_x w_x^{(b)} b_x = \sum_x w_x^{(c)} c_x + c_0$, where constants $\{w_x^{(a)}, w_x^{(b)}, w_x^{(c)}, c_0\}$ depend solely on the circuit and the public values while $\{a_x, b_x, c_x\}$ are the wire assignments (depends on values known only to the prover, or say witness). Note also that only wires of multiplication gates of intermediate values are labeled. Wires of addition gates and multiplication gates with public inputs (constants) are handled by linear constraints. To prove that the prover knows some input such that the output of the circuit above is some specific number, say 0, one can add another linear constraint as $c_3 = 0$.

Bootle et al. [46] showed how to transform a circuit into constraints, and their zero-knowledge argument system works directly over a set of constraints.

Constraints for Correctness of Paillier Encryption. Our first contribution is a (compact) set of constraints handcrafted to represent correct encryption. The prover wants to prove that he knows a pair, (m, r), satisfying $c = (1+N)^m \cdot r^N \mod N^2$, which can be computed as $c = (1+mN) \cdot r^N \mod N^2$, where N is an RSA modulus. Note that $(1+mN)$ is readily a linear constraint. We focus on our set of constraints for representing r^N.

Let $\alpha = \lceil \log N \rceil$ and $\{n_\alpha, \ldots, n_2, n_1\}$ be the binary decomposition of N. That is, $N = 2^{\alpha-1} \cdot n_\alpha + \cdots + 2 \cdot n_2 + n_1$. Define a sequence $\tilde{R} = \{\tilde{R}_1 = r, \tilde{R}_2 = r^2, \ldots, \tilde{R}_\alpha = r^{2^{\alpha-1}}\}$. Let β be the hamming weight of N. Define an index set $\tilde{D} = \{\gamma | n_\gamma = 1\}$, including all index positions of 1's in the decomposition of N. We have $|\tilde{D}| = \beta$. We use $d_1, \ldots, d_\beta$ to denote elements of $\tilde{D}$ with $d_i < d_j$ if $i < j$.

Define sequence $\tilde{S} = \{\tilde{S}_1, \ldots, \tilde{S}_\beta\}$ such that $\tilde{S}_1 = \tilde{R}_{d_1}$, $\tilde{S}_k = \tilde{S}_{k-1} \cdot \tilde{R}_{d_k}$ for $k \in [2, \beta]$. We have $\tilde{S}_\beta = r^N$ since $\tilde{R}_1 = r$. Note that $\tilde{R}$, $\tilde{S}$ are the intermediate values when we calculate r^N from r using the square-and-multiply algorithm. One can uniquely compute $(\alpha, \beta, \tilde{D})$ from N.

As an example, consider N = 11, which can be represented as 1011 in binary. Then $\alpha = 4$ and $\beta = 3$. To compute r^{11}, the sequence $\tilde{R}$ is set as, $\tilde{R} = \{\tilde{R}_1 = r, \tilde{R}_2 = r^2, \tilde{R}_3 = r^4, \tilde{R}_4 = r^8\}$. The sequence $\tilde{D}$ is, $\tilde{D} = \{d_1 = 1, d_2 = 2, d_3 = 4\}$. We have $\tilde{S} = \{\tilde{S}_1 = \tilde{R}_1 = r, \tilde{S}_2 = \tilde{S}_1 \cdot \tilde{R}_2 = r^3, \tilde{S}_3 = \tilde{S}_2 \cdot \tilde{R}_4 = r^{11}\}$.

The correctness of $\tilde{R}$ and $\tilde{S}$ implies that r^N is computed correctly. Thus, correct encryption of Paillier ciphertext can be represented using the following constraints.

$$\begin{array}{ll} c = \dot{T} * \tilde{S}_\beta & \\ \dot{T} = 1 + m * N & \\ \tilde{R}_1 = r \quad \text{//for clarity} & \\ \tilde{R}_{i+1} = \tilde{R}_i * \tilde{R}_i & i \in [1, \alpha - 1] \\ \tilde{S}_1 = \tilde{R}_{d_1} \quad \text{//for clarity} & \\ \tilde{S}_k = \tilde{S}_{k-1} * \tilde{R}_{d_k} & k \in [2, \beta] \end{array} \tag{1}$$

For ease of writing, we define the above set of constraints as $\mathsf{Const}_{\{c,m,r\}}$, with respect to c, m, and r. Note that $\tilde{R}_1$, r (resp. $\tilde{S}_1$, $\tilde{R}_{d_1}$) can be combined into one witness. Thus, $\mathsf{Const}_{\{c,m,r\}}$ contains $\alpha + \beta - 1$ multiplication constraints and one linear constraint.

Proof that A Message Is Binary. Very often, we need to prove that a variable in the constraint is binary. For example, we need to prove that c is the encryption of a binary message m. If the constraints are defined over a prime field, adding the following constraint is sufficient:

$$m * (m-1) = 0.$$

However, we work in $\mathbb{Z}_{N^2}$ and the above constraint does not guarantee that m is binary. According to the Chinese Reminder Theorem, there are 4 values satisfying this constraint:

$$\begin{cases} m = & 0 \\ m = & 1 \\ m = & q^2 \cdot [(q^2)^{-1} \mod p^2] \; (= X) \\ m = & p^2 \cdot [(p^2)^{-1} \mod q^2] \; (= Y), \end{cases}$$

One of the core technical contributions of this work is an innovative statistical argument to ensure that m is binary. Specifically, our solution requires that the prover commits a random "noise" value, R', chosen from a relatively small range $\mathcal{L}$[5], say, $\mathcal{L} := \{1, \ldots, 2^{256}\}$. The verifier chooses a random challenge, $\ell \in \{0, 1\}$, and the prover is required to give $L' := \ell m + R'$, along with a proof that L' is computed correctly. For simplicity, we will also use Paillier encryption for the "commitment" of R'. The proof that L' is correctly computed can be represented by a linear constraint.

More concretely, the prover computes and sends $c' = (1+N)^{R'} * r'^N \mod N^2$ to the verifier, who replies with a challenge bit ℓ, and the prover sends L' along with a proof that the following constraints are satisfied:

$$\begin{aligned} &\mathsf{Const}_{\{c,m,r\}} \\ &m * (m-1) = 0 \\ &\ell * m + R' = L' \\ &\mathsf{Const}_{\{c',R',r'\}}. \end{aligned}$$

Besides checking the proof, the verifier also checks whether $L' \leq \max\{\mathcal{L}\} + 1$. For an honest prover, a statistical argument ensures L' leaks negligible information about m, since R' is much larger than m. A cheating prover may use X or Y as a witness. Recall that X and Y are large (on the order of p^2 or q^2), the only way for a cheating prover to ensure $L' := \ell * m + R' \leq \max\{\mathcal{L}\} + 1$ is to guess ℓ and pick R' accordingly. If he/she guesses $\ell = 1$, he/she should pick a large R' such that X (or Y) plus R' modulo N^2 is within the expected range. Likewise, if he/she guesses $\ell = 0$, he/she should pick a small R', i.e., $R' \leq \max\{\mathcal{L}\}+1$. Therefore, with probability $1/2$, a cheating prover will be caught. To amplify soundness, the above process could be repeated κ times (say $\kappa = 128$).

Proof that Polynomially Many Messages are Binary. Our method can be extended to prove that polynomially many witnesses are binary. Specifically, assume we would like to prove that there are $\mathcal{N}_p$ ciphertexts, each of which encrypts a binary message. We use (m_i, c_i) to denote one message-ciphertext pair, where $i \in [1, \mathcal{N}_p]$. Same as before, the auxiliary information generated by the prover is $\{R'_j\}_{j\in[1,\kappa]}$, encrypted in $\{c'_j\}_{j\in[1,\kappa]}$ using randomness $\{r'_j\}_{j\in[1,\kappa]}$. Now, the random challenges from the verifier are $\{\ell_j^{(i)}\}_{i\in[1,\mathcal{N}_p],j\in[1,\kappa]}$. The corresponding constraints are:

$$\begin{aligned} &\mathsf{Const}_{\{c_i,m_i,r_i\}}, \; i \in [1, \mathcal{N}_p] \\ &m_i * (m_i - 1) = 0, \; i \in [1, \mathcal{N}_p] \\ &\sum_{i=1}^{\mathcal{N}_p} \ell_j^{(i)} * m_i + R'_j = L'_j, \; j \in [1, \kappa] \\ &\mathsf{Const}_{\{c'_j,R'_j,r'_j\}}, \; j \in [1, \kappa]. \end{aligned}$$

Same as above, in addition to checking the proof, the verifier checks whether $L'_j \leq \max\{\mathcal{L}\} + \mathcal{N}_p$ for each $j \in [1, \kappa]$. We would like to remark that the amortized cost for proving one message, say m, being binary is 1 (i.e., satisfying the constraint $m * (m-1) = 0$). Intuitively, if any of the m_i is malformed (say, $m_i = X$ or $m_i = Y$), the probability that all $\{L'_j\}$ satisfy, $L'_j \leq \max\{\mathcal{L}\} + \mathcal{N}_p$, is $2^{-\kappa}$, which is negligible when we set κ to 128.

The actual analysis is much more involved since we need to show no matter how a cheating prover chooses his m_i's, the probability that it can pass the verification is bounded (in fact, we show that it is at most $1/2$) if any of the m_i is X or Y (and is independent of the number of messages). The analysis is shown in Lemma 6.1.

Proof of Messages with Correct Structure. Recall our goal is to prove the correctness of encryption for messages with specific formats such as packing. For example, we may consider packing two binary messages into one ciphertext, where the first two slots are 32-bit. That is,

$$m = \underbrace{00\ldots0}_{32-bit}\ldots\underbrace{00\ldots b_{32}}_{32-bit}\underbrace{00\ldots b_0}_{32-bit}.$$

We can make use of the constraint $2^{32} \cdot b_2 + b_1 = m$ to shift the bits to the correct position. For instance, the following set of constraints represents all $\mathcal{N}_p$ ciphertexts encrypt 2 bits, each occupying a 32-bit slot:

$$\begin{aligned} &\mathsf{Const}_{\{c_i,m_i,r_i\}} \\ &2^{32} \cdot b_{32}^{(i)} + b_0^{(i)} = m_i \\ &b_0^{(i)} * (b_0^{(i)} - 1) = 0 \\ &b_{32}^{(i)} * (b_{32}^{(i)} - 1) = 0 \\ &\mathsf{Const}_{\{c'_j,R'_j,r'_j\}} \\ &\sum \left(l_{j,0}^{(i)} b_0^{(i)} + l_{j,32}^{(i)} b_{32}^{(i)} \right) + R'_j = L'_j, \end{aligned}$$

for $i \in [1, \mathcal{N}_p]$ and $j \in [1, \kappa]$.

One may wish to directly extend the above method to support Paillier with packing for an arbitrary number of slots and plaintexts, e.g.,

$$m_i = \underbrace{0\ldots0b_{32\cdot63}^{(i)}}_{32-bit}\underbrace{0\ldots0b_{32}^{(i)}}_{32-bit}\ldots\underbrace{0\ldots0b_0^{(i)}}_{32-bit},$$

where $|N| = 2048$, and we pack 64 bits into 64 slots. However, our analysis showed that this is not straightforward. The reason is that we have to ensure "bits" $\{b_{32\cdot63}^{(i)}, \ldots, b_0^{(i)}\}$ and auxiliary input $\{R'_j\}$ are fixed before random challenges $\{l_{j,32(s-1)}^{(i)}\}$ are chosen. However, given (1) $c_i =$

5. Here, we also require the prover to attach a range proof for R'.

$(1+m_iN)r_i^N \bmod N^2$; (2) $m_i = \sum_s 2^{32(s-1)} b_{32(s-1)}^{(i)} \bmod N^2$; and (3) $b_{32(s-1)}^{(i)} * (b_{32(s-1)}^{(i)} - 1) = 0 \bmod N^2$, the set $\{b_{32(s-1)}^{(i)}\}$ is not unique (despite m is fixed due to the injective nature of encryption).

This counter-intuitive observation arises from the fact that at this point we cannot ensure $\{b_{32(s-1)}^{(i)}\}$'s are binary and thus $m = \sum_s 2^{32(s-1)} b_{32(s-1)}^{(i)} \bmod N^2$ may have multiple solutions. There is a possibility that a malicious prover may choose different $\{b_{32(s-1)}^{(i)}\}$ after seeing challenges, and the analysis in Lemma 6.1 crucially relies on the fact that the prover's "bits" are fixed before seeing these challenges.

We tackle this subtlety by carefully identifying the condition under which the prover's "bits" are fixed. Specifically, we observe that if message m satisfies $|m| < \min\{|p|, |q|\}$ (where p and q is the factorization of N), fulfilling constraints (1) $m = \sum_s 2^{32(s-1)} b_{32(s-1)}$; and (2) $b_{32(s-1)} * (b_{32(s-1)} - 1) = 0$ (for $s \in [1, 64]$), then the set $\{b_{32(s-1)}\}$ is unique. The formal analysis is shown in Lemma 6.2.

Since under this condition the set $\{b_{32(s-1)}\}$ satisfying the constraints is unique, the "bits" are fixed given c and above constraints. Consequently, when the message space is 2048-bit, it is only safe to use 1024 bits. In other words, we can only use 32 32-bit slots to achieve provable security. This is not ideal and we describe our final solution below.

Our Final Solution. We construct auxiliary messages to fulfill the above "length requirement". Assuming $|N| = 2048$ and we divide the message space into 64 slots (each of which is 32-bit), we need to introduce one new auxiliary message m_t^* for every 15 messages. Thus there will be $\mathcal{N}_p/15$ auxiliary messages in total. Here we give an example to see how we construct m_t^* from messages m_{15t-14} to m_{15t} ($t \in [1, \mathcal{N}_p/15]$),

$$m_t^* \;=\; \underbrace{b_{32\cdot 63}^{(15t)} \dots b_0^{(15t)}}_{\textit{from } m_{15t}} \;\dots\; \underbrace{b_{32\cdot 63}^{(15t-14)} \dots b_0^{(15t-14)}}_{\textit{from } m_{15t-14}},$$

where $b_{32(s-1)}^{(i)}$ indicates the last bit in the s-th slot of message m_i, for $s \in [1, 64]$ and $i \in [1, \mathcal{N}_p]$. Each auxiliary message is 960-bit, and they satisfy,

$$(2^{959} * b_{32\cdot 63}^{(15t)} + \ldots + 2^{896} * b_0^{(15t)}) + \ldots$$
$$+ (2^{63} * b_{32\cdot 63}^{(15t-14)} + \ldots + b_0^{(15t-14)}) = m_t^*,$$

for $t \in [1, \mathcal{N}_p/15]$. We use c_t^* to denote Paillier ciphertexts of m_t^*. As the length of m_t^* satisfies above requirement, we can use our proposed method to prove that all $\{b_{32(s-1)}^{(i)}\}$'s are 0 or 1. Constraints in our final solution are given below:

$$\begin{aligned}
&\mathsf{Const}_{\{c_i, m_i, r_i\}} \\
&\sum_s 2^{32(s-1)} b_{32(s-1)}^{(i)} = m_i \\
&b_{32(s-1)}^{(i)} * (b_{32(s-1)}^{(i)} - 1) = 0 \\
&\mathsf{Const}_{\{c_j', R_j', r_j'\}} \\
&\sum_{i,s} \ell_{j,32(s-1)}^{(i)} b_{32(s-1)}^{(i)} + R_j' = L_j' \\
&\sum_s \sum_k 2^{32(k-1)+s-1} \cdot b_{32(s-1)}^{15(t-1)+k} = m_t^* \\
&\mathsf{Const}_{\{c_t^*, m_t^*, r_t^*\}},
\end{aligned}$$

where $j \in [1, \kappa]$, $s \in [1, 64]$, $k \in [1, 15]$, $i \in [1, \mathcal{N}_p]$ and $t \in [1, \frac{\mathcal{N}_p}{15}]$. We use $\ell_{j,32(s-1)}^{(i)}$ to indicate the random challenges. Since there are more bits now in the computation of L_j', range $\mathcal{L}$ will be slightly enlarged, say, $\mathcal{L} := \{0, 2^{281}\}$.

3. Preliminaries

In what follows, we use an adversary $\mathcal{A}$ as an interactive probabilistic polynomial time Turing Machine, whose running time is polynomial in the security parameter λ.

Assumptions. Let Setup be an algorithm outputting $(\mathbb{G}, N^2, g)$, with input 1^λ. $\mathbb{G}$ is the description of a finite cyclic group with composite order N^2, where $N = pq$ is a RSA modulus, $|N^2| = \lambda$ and g is the generator[6].

Definition 3.1. (Discrete Logarithm Relation Assumption on Composite Group). *This assumption holds if for all $n \geq 1$ and non-uniform PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c|c} \exists a_i \neq 0 & (\mathbb{G}, N^2, g) \leftarrow \mathsf{Setup}(1^\lambda), \\ \text{and} & g_1, \ldots, g_n \leftarrow \mathbb{G}\ , \\ g^{a_0}\prod_{i=1}^n g_i^{a_i} = 1 & a_0, \ldots, a_n \leftarrow \mathcal{A}(\mathbb{G}, N^2, g, \{g_i\}_i) \end{array}\right] \approx 0,$$

relative to Setup. *We say $g^{a_0}\prod_{i=1}^n g_i^{a_i} = 1$ a non-trivial discrete log relation between $g_1, \ldots, g_n$. It is known that it is equivalent to the discrete logarithm assumption.*

Pedersen Commitment. In our protocol, we require a Pedersen commitment scheme, which works over the group $\mathbb{G}$ of Definition 3.1. Informally, it contains 2 polynomial algorithms $\mathsf{PDC} = (\mathsf{PDC.Gen}, \mathsf{PDC.Com})$ where the commitment has the form $ct = g^m \cdot h^r$. We use a variant, that allows to commit to multiple values at once. Its formal definition is given in Appendix A.

Paillier Encrption. The Paillier encryption consists of three polynomial algorithms, $\mathsf{PL} = (\mathsf{PL.Gen}, \mathsf{PL.Enc}, \mathsf{PL.Dec})$. Its formal definition is deferred in Appendix A.

4. Our Main Protocol

We present the constructions of our main protocol ZKAoK^* and its extension, ZKAoK^+.[7] Since the ZKP system proposed in [46] requires constraints as inputs, it is sufficient for us to specify constraints for corresponding relations under modulo N^2. It is straightforward to adapt [46] to work over $\mathbb{Z}_{N^2}$ except how the cyclic group with order N^2 can be generated. Here we describe one potential method. Given N^2, one first finds a prime Q, s.t., $Q = fN^2 + 1$ for some small integer f. Then, choose an arbitrary element g in $\mathbb{Z}_Q^*$, s.t., $g^{N^2} = 1\ mod\ Q$. We use g to generate $\mathbb{G}$. Besides, we provide a discussion of how to prove other Paillier plaintext relations in Appendix C.

In our protocols, the verifier should also require the prover to prove that $N = pq$ is a valid RSA modulus, where p and q are large primes. This one-time setup can be implemented using existing methodologies [75], [79]. Further elaboration is provided in Subsection 1.7.

6. To find such a group $\mathbb{G}$, one can use methods specified in Sec. 4

7. Looking ahead, our three protocols are specifically designed for 3 requirements presented in Section 1.2 respectively.

4.1. Constraints for A Valid Paillier Message Ciphertext Pair

We recall a building block, $\mathsf{Const}_{\{c,m,r\}}$, that specifies constraints for proving a valid Paillier message-ciphertext pair (m, c) with randomness r. Let $\alpha = \lceil \log N \rceil$ and $S_N = \{n_\alpha, \ldots, n_2, n_1\}$, the set containing the binary decomposition bits of N, satisfying $N = \sum_{k\in[1,\alpha]} 2^{k-1} \cdot n_k$. Define the sequence $\tilde{R} := \{\tilde{R}_k : \forall k \in [1, \alpha], \tilde{R}_k = r^{2^{k-1}}\}$. Let β be the hamming weight of N. Define the index sequence as,

$$\tilde{D} := \{d_\gamma | \forall \gamma \in [1, \beta],\ n_{d_\gamma} = 1\ \wedge\ n_{d_\gamma} \in S_N \\ \wedge\ \forall \gamma \in [1, \beta - 1], d_\gamma < d_{\gamma+1}\}.$$

We have $|\tilde{D}| = \beta$. Define sequence,

$$\tilde{S} := \{\tilde{S}_k | \forall k \in [1, \beta], \tilde{S}_k = \tilde{S}_{k-1} \cdot \tilde{R}_{d_k}\ \wedge\ \tilde{S}_0 = 1\},$$

where $|\tilde{S}| = \beta$ and $\tilde{S}_\beta = r^N$ (with setting $\tilde{R}_1 = r$). We denote the constraints for proving that c is a valid Paillier ciphertext of m with r as $\mathsf{Const}_{\{m,c,r\}}$, the same as Equ. 1.

4.2. Our Main Protocol ZKAoK*

Relation $\mathcal{R}^*$ for Main Protocol. Our goal is to prove that given $\mathcal{N}_p$ ciphertexts, $\{c_i\}_{i\in[1,\mathcal{N}_p]}$, where c_i is the encryption of m_i satisfies,

$$m_i = \underbrace{0\ldots 0 b^{(i)}_{32\cdot(64-1)}}_{32-bit}\underbrace{0\ldots 0 b^{(i)}_{32\cdot(63-1)}}_{32-bit}\cdots\underbrace{0\ldots 0 b^{(i)}_{0}}_{32-bit}. \quad (2)$$

That is to say, each message contains 64 32-*bit* slots, and all bits except the last bit in each slot are 0. (One can easily prove that each message contains 32 32-*bit* slots when $|N| = 1024$ using the same technique with $s \in [1, 32]$.) Formally, the relation we prove can be described by $\mathcal{R}^*$ defined below:

$$\mathcal{R}^* = \{(\{c_i\}_{i\in[1,\mathcal{N}_p]}, N), (m_i, r_i, b^{(i)}_{32(s-1)})_{i\in[1,\mathcal{N}_p],s\in[1,64]} : \\ \forall\ i \in [1, \mathcal{N}_p]\ \textit{and}\ s \in [1, 64], \\ m_i = \textstyle\sum_{s\in[1,64]} 2^{32(s-1)} \cdot b^{(i)}_{32(s-1)} \mod N \\ \wedge\ b^{(i)}_{32(s-1)} \in \{0, 1\} \wedge c_i = (1 + N)^{m_i} \cdot r_i^N \mod N^2\}. \quad (3)$$

Main Protocol ZKAoK*. We define,

$$m^*_t \ = \ \underbrace{b^{(15t)}_{32\cdot 31}\ldots b^{(15t)}_0}_{\textit{from}\ m_{15t}} \ \cdots \ \underbrace{b^{(15t-14)}_{32\cdot 31}\ldots b^{(15t-14)}_0}_{\textit{from}\ m_{15t-14}}.$$

Then m^*_t satisfies,

$$m^*_t = 2^{959} \cdot b^{(15t)}_{64} + \cdots + b^{(15t-14)}_1 \mod N^2 \\ = \sum_{s\in[1,64],k\in[1,15]} 2^{64(k-1)+s-1} \cdot b^{(15(t-1)+k)}_s \mod N^2,$$

where $t \in [1, \mathcal{N}_p/15]$. We also define c^*_t as its encryption,

$$c^*_t = (1 + N)^{m^*_t} \cdot r^N_t \mod N^2.$$

We show our protocol, ZKAoK*, in Fig. 2. We take $\mathcal{L} = [0, 2^{281}]$. In addition, the prover should also prove that each R'_j is chosen from $\mathcal{L}$. This can be done by a standard range proof or ZKAoK′, which will be discussed in Section 5.

4.3. An Extended Protocol ZKAoK+

Relation $\mathcal{R}^+$. We now extend our main protocol to ZKAoK+, proving that structured messages satisfy additional requirements (i.e., sum of records and sum of entities). Its formal relation $\mathcal{R}^+$ is,

$$\mathcal{R}^+ = \{(\{c_i\}_{i\in[1,\mathcal{N}_p]}, N, t, T), (m_i, r_i, b^{(i)}_{32(s-1)})_{i\in[1,\mathcal{N}_p],s\in[1,64]} : \\ \forall\ i \in [1, \mathcal{N}_p]\ \textit{and}\ s \in [1, 64], c_i = (1 + N)^{m_i} \cdot r_i^N \mod N^2 \\ \wedge\ m_i = \textstyle\sum_{s\in[1,64]} 2^{32(s-1)} \cdot b^{(i)}_{32(s-1)} \mod N \wedge\ b^{(i)}_{32(s-1)} \in \{0, 1\} \\ \wedge\ \textstyle\sum_{s\in[1,64]} b^{(i)}_{32(s-1)} \le t\ \wedge \textstyle\sum_{i\in[1,\mathcal{N}_p]} b^{(i)}_{32(s-1)} \ge T\}. \quad (4)$$

Our Solution. The sum of records property requires that there are at most t 1's packed in $\{b^{(i)}_{32(s-1)}\}_{s\in[1,64]}$ for each message m_i. Suppose $t = 2^x$. We define u_i as the summation of all $\{b^{(i)}_{32(s-1)}\}$'s towards the same entity i. It is sufficient to prove that each u_i can be decomposed by x bits, $\{w^{(i)}_x, \ldots, w^{(i)}_2, w^{(i)}_1\}$, s.t., $u_i = \sum_{s\in[1,64]} b^{(i)}_{32(s-1)} = \sum_{v\in[1,x]} 2^{v-1} w^{(i)}_v$, where $w^{(i)}_v \in \{0, 1\}$. We define c_{u_i} as the Paillier encryption of u_i with randomness r_{u_i}.

The sum of units property requires that there are at least T 1's among $\{b^{(i)}_{32(s-1)}\}_{i\in[1,\mathcal{N}_p]}$ towards each unit s. Assume $T = 2^y$. We use ψ_s to denote the summation of binary records towards the same s as, $\psi_s = \sum_{i\in[1,\mathcal{N}_p]} b^{(i)}_{32(s-1)}$. Suppose ψ_s can be decomposed into n binary bits, $\{\psi^{(s)}_n, \ldots, \psi^{(s)}_2, \psi^{(s)}_1\}$, s.t, $\psi_s = \sum_{\rho\in[1,n]} 2^{\rho-1} \cdot \psi^{(s)}_\rho$. Define ϕ_s as the summation of decomposition bits from $\psi^{(s)}_n$ to $\psi^{(s)}_{y+1}$ as, $\phi_s = \sum_{k\in[y+1,n]} \psi^{(s)}_k$. It is sufficient to prove that for every $s \in [1, 64]$, there exists an integer γ_s s.t., $\phi_s \cdot \gamma_s = \theta_s$, where θ_s is a random integer challenge.

Besides, it is required to prove that all the decomposition bits $(\{w^{(i)}_v\}_{i\in[1,\mathcal{N}_p],v\in[1,x]}, \{\psi^{(s)}_k\}_{k\in[y+1,n],s\in[1,64]})$ are bits. We employ the same technique as in the main protocol. To fix $\{w^{(i)}_v\}$, one needs to further encrypt its summation u_i to c_{u_i}. $\{\psi^{(s)}_k\}$ doesn't need auxiliary encryption as the verifier can compute $\prod_i c_i$ and parse the encryption of ψ_s itself. Furthermore, one will need the statistical argument to prevent a cheating prover. We re-use the statistical argument in ZKAoK*. We define constraints $\mathcal{C}'_{\mathcal{R}^*}$ by changing original constraint $L'_j = \sum_{i,s} \ell^{(i)}_{j,32(s-1)} b^{(i)}_{32(s-1)} + R'_j$ into,

$$L'_j = \textstyle\sum_{i,s,v,k} \ell^{(i)}_{j,32(s-1)} b^{(i)}_{32(s-1)} + \ell'^{(i)}_v w^{(i)}_v + \ell'^{(s)}_k \psi^{(s)}_k + R'_j,$$

where $\{\ell^{(i)}_{j,32(s-1)}\}$, $\{\ell'^{(i)}_v\}$, and $\{\ell'^{(s)}_k\}$ are randomly chosen from $\{0, 1\}$ by the verifier and $i \in [1, \mathcal{N}_p]$, $s \in [1, 64]$, $v \in [1, x]$, $k \in [y + 1, n]$, and $j \in [1, \kappa]$. Then $\mathcal{C}_{\mathcal{R}^+}$ is,

$$\begin{cases} \mathcal{C}'_{\mathcal{R}^*} & \\ \mathsf{Const}_{\{c_{u_i}, u_i, r_{u_i}\}} & \forall i \in [1, \mathcal{N}_p] \\ u_i = \sum_{s=1}^{64} b^{(i)}_{32(s-1)} & \forall i \in [1, \mathcal{N}_p] \\ u_i = \sum_{v\in[1,x]} 2^{v-1} w^{(i)}_v & \forall i \in [1, \mathcal{N}_p] \\ w^{(i)}_v (w^{(i)}_v - 1) = 0 & \forall v \in [1, x], \forall i \in [1, \mathcal{N}_p] \\ \psi_s = \sum_{i\in[1,\mathcal{N}_p]} b^{(i)}_{32(s-1)} & s \in [1, 64] \\ \psi_s = \sum_{\rho\in[1,n]} 2^{n-1} \cdot \psi^{(s)}_\rho & s \in [1, 64] \\ \phi_s = \sum_{k\in[y+1,n]} \psi^{(s)}_k & s \in [1, 64] \\ \psi^{(s)}_k (\psi^{(s)}_k - 1) = 0 & \forall k \in [y + 1, n], \forall s \in [1, 64] \\ \phi_s \cdot \gamma_s = \theta_s & s \in [1, 64], \end{cases} \quad (5)$$

$\mathcal{P} : (N, \{(c_i, m_i, r_i, b^{(i)}_{32(s-1)})\}_{i\in[1,\mathcal{N}_p], s\in[1,64]})$ $\qquad$ $\mathcal{V} : (N, \{c_i\}_{i\in[1,\mathcal{N}_p]})$

$\forall j \in [1, \kappa],\ R'_j \leftarrow \mathcal{L}$
$\forall t \in [1, \mathcal{N}_p/15]$, *construct auxiliary messages* m^*_t
$\forall j \in [1, \kappa]$, *compute* $c'_j = \mathsf{PL.Enc}_{\mathsf{pk}}(R'_j; r'_j)$
$\forall t \in [1, \mathcal{N}_p/15]$, *compute* $c^*_t = \mathsf{PL.Enc}_{\mathsf{pk}}(m^*_t; r^*_t)$

$$\xrightarrow{(\{c'_j\}_{j\in[1,\kappa]}, \{c^*_t\}_{t\in[1,\mathcal{N}_p/15]})}$$

$\forall i \in [1, \mathcal{N}_p]$,
$\forall s \in [1, 64]$,
$\forall j \in [1, \kappa]$,
$l^{(i)}_{j,32(s-1)} \leftarrow \{0, 1\}$

$$\xleftarrow{\{l^{(i)}_{j,32(s-1)}\}_{i\in[1,\mathcal{N}_p], s\in[1,64], j\in[1,\kappa]}}$$

$\forall j \in [1, \kappa]$, *compute,*
$L'_j = \sum_{i=1}^{\mathcal{N}_p} \sum_{s=1}^{64} l^{(i)}_{j,32(s-1)} b^{(i)}_{32(s-1)} + R'_j$

$$\xrightarrow{\{L'_j\}_{j\in[1,\kappa]}}$$

$\forall j \in [1, \kappa]$,
checks $L'_j \le \max\{R'_j\} + 64\mathcal{N}_p$?
If yes, continue;
Otherwise, reject.

compute constraints $\mathcal{C}_{\mathcal{R}^*}$ *as,*

$$\begin{cases} \forall i \in [1, \mathcal{N}_p], & \mathsf{Const}_{\{c_i, m_i, r_i\}} \\ \forall j \in [1, \kappa], & \mathsf{Const}_{\{c'_j, R'_j, r'_j\}} \\ \forall t \in [1, \mathcal{N}_p/15], & \mathsf{Const}_{\{c^*_t, m^*_t, r^*_t\}} \\ \forall i \in [1, \mathcal{N}_p], & m_i = \sum_{s=1}^{64} 2^{32(s-1)} b^{(i)}_{32(s-1)} \\ \forall s \in [1, 64], \forall i \in [1, \mathcal{N}_p], & b^{(i)}_{32(s-1)} \cdot (b^{(i)}_{32(s-1)} - 1) = 0 \\ \forall j \in [1, \kappa], & L'_j = \sum_{i,s} \ell^{(i)}_{j,32(s-1)} b^{(i)}_{32(s-1)} + R'_j \\ \forall t \in [1, \mathcal{N}_p/15], & m^*_t = \sum_{s=1}^{64} \sum_{k=1}^{15} 2^{64(k-1)+s-1} \cdot b^{15(t-1)+k}_{32(s-1)} \end{cases}$$

$$\xleftrightarrow{\mathsf{ZKAoK}\ \textit{with inputs}\ \mathcal{C}_{\mathcal{R}^*}\ \textit{over}\ \mathbb{Z}_{N^2}}$$

Figure 2. Our Main Protocol ZKAoK^* for Relation $\mathcal{R}^*$

ZKAoK^+ runs the same as our main protocol, based on the above constraints. The verifier chooses $\{\theta_s\}$ along with $\{\ell^{(i)}_{j,32(s-1)}\}$, $\{\ell'^{(i)}_v\}$, and $\{\ell'^{(s)}_k\}$. It is noted that we only give a naive solution towards $\mathcal{R}^+$ and additional optimizations are possible. One can easily optimize it by encrypting several $\{u_i\}$'s into one message, as each u_i contains at most 64 bits while a Paillier plaintext is 2048 bits when $|N| = 2048$.

5. A Range Proof Protocol

We further construct a range proof protocol ZKAoK'. Specifically, we are going to prove that polynomially many plaintexts $\{m_i\}$ are in the same range, say, $[0, 2^{256}]$, the relation $\mathcal{R}'$ is set as follows,

$$\mathcal{R}' = \{(\{c_i\}_{i\in[1,\mathcal{N}_p]}, N, 2^{256}), (m_i, r_i, b^{(i)}_k)_{i\in[1,\mathcal{N}_p], k\in[1,256]} : \forall\, i \in [1, \mathcal{N}_p],\ c_i = (1+N)^{m_i} \cdot r_i^N \mod N^2 \wedge m_i \le 2^{256}\}. \tag{6}$$

We set $|N| = 2048$ where $|p| = |q| = 1024$. We use $\{b^{(i)}_{256}, \dots, b^{(i)}_2, b^{(i)}_1\}$ to denote the decomposition of m_i. To prove $\mathcal{R}'$ with the above setting, it is sufficient to prove that 1) the decomposition elements of m_i are all 0 or 1; 2) These 256 bits can re-construct m_i; 3) c_i is a valid ciphertext for m_i. One thing that should be addressed is that when proving 256-bit messages satisfy the length requirement, $|m_i| < \min\{|p|, |q|\}$, there is no need to construct auxiliary messages m^*_t. Then the constraints in $\mathcal{C}'$ are,

$$\begin{cases} \mathsf{Const}_{\{c_i, m_i, r_i\}},\ \forall i \in [1, \mathcal{N}_p] \\ \mathsf{Const}_{\{c'_j, R'_j, r'_j\}},\ \forall j \in [1, \kappa] \\ b^{(i)}_{k-1} \cdot (b^{(i)}_{k-1} - 1) = 0,\ \forall i \in [1, \mathcal{N}_p], \forall k \in [1, 256] \\ L'_j = \sum_{k=1}^{256} \sum_{i=1}^{\mathcal{N}_p} l^{(i)}_{j,k-1} \cdot b^{(i)}_{k-1} + R'_j,\ \forall j \in [1, \kappa] \\ m_i = \sum_{k=1}^{256} 2^{k-1} \cdot b^{(i)}_{k-1},\ \forall i \in [1, \mathcal{N}_p], \end{cases} \tag{7}$$

where $l^{(i)}_{j,k-1} \leftarrow \{0, 1\}$, is randomly chosen from the verifier for each $b^{(s)}_{k-1}$ and $R'_j \leftarrow \mathcal{L}$. We can set $\mathcal{L} = [0, 2^{281}]$ to hide bits when having millions of messages in the above setting. Besides, same as our main protocol, the prover should send $\{c'_j\}$ before the verifier chooses challenges $\{l^{(i)}_{j,k}\}$.

6. Security Proof

In this section, we analyze the security of our main protocol, as others are similar to analyze. It is easy to verify the completeness of the protocol. Also, special honest verifier zero-knowledge property comes from the underlying zero-knowledge argument system and security of the Paillier encryption scheme. Next, we argue the special soundness of our protocol. It is sufficient to show that $\mathcal{R}^*$ is equivalent to the constraints specified by $\mathcal{C}_{\mathcal{R}^*}$ in Fig. 2 . Here, we focus on showing that if $\mathcal{C}_{\mathcal{R}^*}$ holds, then each $b^{(i)}_{32(s-1)}$ is from $\{0,1\}$ with all but negligible probability.

Let $\bar{q} = q^{-1} \mod p$, $\bar{p} = p^{-1} \mod q$, then $b^{(i)}_{32(s-1)} = b^{(i)}_{32(s-1)} \cdot b^{(i)}_{32(s-1)}$ implies that $b^{(i)}_{32(s-1)} \in \{0, 1, q \cdot \bar{q}, p \cdot \bar{p}\}$. Note that both $q \cdot \bar{q}$ and $p \cdot \bar{p}$ are much larger than p, q, with high probability, thus, it is sufficient to show that $b^{(i)}_{32(s-1)}$ is small. We complete this task by revealing a random subset sum of all $b^{(i)}_{32(s-1)}$. If any $b^{(i)}_{32(s-1)}$ is large, then at least half of the subset sums will be large (as explained in Lemma 6.1). Thus, we can bound $b^{(i)}_{32(s-1)}$ via showing that random subset sums of all $b^{(i)}_{32(s-1)}$ are always small. One subtle issue here is that a (malicious) prover may use different $b^{(i)}_{32(s-1)}$ to answer different challenges. We solve this issue by committing all $b^{(i)}_{32(s-1)}$ in the beginning into $\{c^*_t\}$, which was proved in Lemma 6.2. Specifically, we formalize the above proof idea in Theorem 6.1 and corresponding lemmas.

Theorem 6.1. *If the condition $\mathcal{N}_b + \max\{R'_j\} < \min\{X, Y\} < N^2 - \mathcal{N}_b - \max\{R'_j\}$ holds, the argument presented in our main protocol (ZKAoK^*) using the protocol in [46] for relation $\mathcal{R}^*$ satisfies perfect completeness, statistical special honest verifier zero-knowledge and statistical witness-extended emulation.*

The proof of Theorem 6.1 is given in Appendix D.

Lemma 6.1. *Let B be a set $\{b_1, \ldots, b_k\}$, where $b_j \in \{0, 1, X, Y\}$ for $1 \le j \le k$. Let $L = b_1 \cdot l_1 + b_2 \cdot l_2 + \cdots + b_k \cdot l_k + R \mod N^2$, where $l_j \leftarrow \{0,1\}$, $R \leftarrow \mathcal{L}$ and $1 \le j \le k$. Define the event that there exists a j such that $b_j \notin \{0,1\}$ as* ***NonBits****; Otherwise, define the event as* ***Bits****. Let $M = k + \max\{R\}$. Assuming that N is a correctly generated RSA modulus and $M < \min\{X, Y\} < N^2 - M$, then,*

$$\Pr[L \le M \mid \textsf{NonBits}] \le \frac{1}{2}.$$

Please note that L is computed under modulo N^2. Then for $1 \le i \le t$, define $L_i = b_1 \cdot l^{(i)}_1 + b_2 \cdot l^{(i)}_2 + \cdots + b_j \cdot l^{(i)}_j + \cdots + b_k \cdot l^{(i)}_k + R_i \mod N^2$, where $l^{(i)}_j \leftarrow \{0,1\}$ and $R_i \leftarrow \mathcal{L}$ is chosen from the same range as R. We have,

$$\Pr[L_1 \le M \ \wedge\ L_2 \le M \ \wedge \cdots \wedge\ L_t \le M \mid \textsf{NonBits}] \le \frac{1}{2^t}.$$

Proof. (of Lemma 6.1) Let $U = \{0,1\}^k$ be the challenge space for random challenges, $l_1, l_2, \ldots, l_k$. Let $M = k + \max\{R\}$. If $\mathsf{NonBits}$ happens, there exists a j, s.t., $b_j = \{X, Y\}$. Let $\{l_1, \ldots, l_j, \ldots, l_k\}$ be challenge bits such that,

$$L = b_1 \cdot l_1 + \cdots + b_j \cdot l_j + \cdots + b_k \cdot l_k + R \le M \mod N^2.$$

Then for another challenge set, $\{l_1, \ldots, \bar{l}_j, \ldots, l_k\}$, we define,

$$\bar{L} = b_1 \cdot l_1 + \cdots + b_j \cdot \bar{l}_j + \cdots + b_k \cdot l_k + R \mod N^2,$$

where $\bar{l}_j$ indicates the reverse of l_j. We have, $\bar{L} = L \pm X(\textit{resp. } Y) > M \mod N^2$, since X (resp. Y) is a big integer. That is, there exists at least half of the challenges in challenge space U satisfying $\bar{L} > M \mod N^2$. Thus, $\Pr[L \le M \mid \mathsf{NonBits}] \le \frac{1}{2}$.

Similarly, for the same set B with $b_j \in \{X, Y\}$, if it satisfies,

$$L_i = b_1 \cdot l^{(i)}_1 + b_2 \cdot l^{(i)}_2 + \cdots + b_j \cdot l^{(i)}_j + \cdots + b_k \cdot l^{(i)}_k + R_i \le M \mod N^2,$$

Then we can construct $\bar{L}_i$ such that,

$$\bar{L}_i = b_1 \cdot l^{(i)}_1 + b_2 \cdot l^{(i)}_2 + \cdots + b_j \cdot \bar{l}^{(i)}_j + \cdots + b_k \cdot l^{(i)}_k + R_i > M \mod N^2,$$

where $R_i \leftarrow \mathcal{L}$ is chosen from the same range as R. Therefore for each $L_i \le M \mod N^2$, we can construct $\bar{L}_i > M \mod N^2$ as before. Similarly, we have,

$$\Pr[L_i \le M \mid \mathsf{NonBits}] \le \frac{1}{2},$$

for all i. As all challenge bits are independently chosen, then the probability that all L_i's satisfy $L_i \le M \mod N^2$ is,

$$\begin{aligned}
&\Pr[L_1 \le M \ \wedge \cdots \wedge\ L_t \le M \mid \mathsf{NonBits}] \\
=&\Pr[L_1 \le M \mid \mathsf{NonBits}] \cdots \Pr[L_t \le M \mid \mathsf{NonBits}] \\
\le&\frac{1}{2} \cdots \frac{1}{2} = \frac{1}{2^t},
\end{aligned}$$

which completes the proof. □

Lemma 6.2. *For every $m < N^2$ with ciphertext c, there exists at most one possible witness set $\{b_0, b_1 \ldots, b_u\}$ satisfying the following constraints,*

$$c = (1+N)^m \cdot r^N \mod N^2 \tag{8}$$
$$m = 2^u \cdot b_u + \cdots + 2 \cdot b_1 + b_0 \mod N^2 \tag{9}$$
$$0 = b_i \cdot (b_i - 1) \text{ for } i \in [0, u] \mod N^2, \tag{10}$$

provided that the following inequality holds,

$$u + 1 < \min\{|p|, |q|\},$$

where $N = pq$.

Proof. (of Lemma 6.2) To prove this lemma, we use contradiction. Assume that there exists two different sets, $\{b_i\}^u_{i=0}$ and $\{b'_i\}^u_{i=0}$ such that there exists a j where $b_j \ne b'_j$, for $0 \le j \le u$, satisfying constraints 8 to 10. Suppose m and m' are two messages reconstructed from these two sets following Constraint 9, with the same encryption c. We denote these two sets as, $(c, m, \{b_i\}^u_{i=0})$ and $(c, m', \{b'_i\}^u_{i=0})$. According to Constraint 8, we get $m = m' + kN \mod N^2$, where $k \ge 0$. Constraint 9 provides,

$$m = 2^u b_u + \cdots + 2b_1 + b_0 \mod N^2 \tag{11}$$
$$m' = 2^u b'_u + \cdots + 2b'_1 + b'_0 \mod N^2. \tag{12}$$

Equation 11 minus Equation 12 yields,

$$d = kN = 2^u(b_u - b'_u) + \cdots + 2(b_1 - b'_1) + b_0 - b'_0 \mod N^2. \quad (13)$$

Consequently, $d = 0 \mod N$. We use $d_i = b_i - b'_i$, to denote the difference (in $\mathbb{Z}$) between each element in $\{b_i\}_{i=0}^u$ and $\{b'_i\}_{i=0}^u$. We get,

$$d = 2^u d_u + \cdots + 2d_1 + d_0 = 0 \mod N. \quad (14)$$

Let $e_i = d_i \mod p$ and $f_i = d_i \mod q$. With $N = pq$, this implies that $d = 0 \mod p$ (*resp.* q). Therefore, considering modulo p,

$$2^u e_u + \cdots + 2e_1 + e_0 = 0 \mod p. \quad (15)$$

According to Constraint 10, we have b_i (*resp.* b'_i) $\in \{0, 1, X, Y\}$. When it goes to modulo a prime number p, they can only be 0 or 1. Thus, we have $e_i \in \{-1, 0, 1\}$. Given $u + 1 < \min\{|p|, |q|\}$, Equation 15 is still satisfied without the final modulo operation, and we have,

$$2^u e_u + \cdots + 2e_1 + e_0 = 0. \quad (16)$$

To satisfy the above equation, $e_i = 0$ where $i \in [0, u]$. Similarly, we can prove that $f_i = 0$. This implies that $d_i = 0 \mod p$ and $d_i = 0 \mod q$. Additionally, since we have b_i (resp. b'_i) $\in \{0, 1, X, Y\}$, it follows that $d_i \in \{0, 1, X, Y, -1, -X, -Y, 1 - X, 1 - Y, X - 1, X - Y, Y - 1, Y - X\}$. Assuming $d_i \neq 0$, we can prove that either $d_i \neq 0 \mod p$ or $d_i \neq 0 \mod q$. We first assume w.l.o.g. that $X = 1 \mod p$, $X = 0 \mod q$, $Y = 0 \mod p$, and $Y = 1 \mod q$. Then if $d_i \in \{1, X, 1 - Y, X - Y\}$, we have $d_i = 1 \mod p$. If $d_i \in \{-1, -X, Y - 1, Y - X\}$, then we have $d_i = -1 \mod p$. If $d_i \in \{1, Y, 1 - X, Y - X\}$ or $d_i \in \{-1, -Y, X - 1, X - Y\}$, we have that $d_i = 1 \mod q$ or $d_i = -1 \mod q$, respectively. As the above contradicts the fact that $d_i = 0 \mod p$ or $d_i = 0 \mod q$, d_i cannot be other values except 0, which completes our proof. $\square$

7. Implementation and Evaluation

To facilitate our explanation, we first recall the notations and their definitions in TABLE 3. We implement our protocol as a proof-of-concept to verify its practicality. Our implementation is written in C++ and we utilize NTL [81], for big integer operations. Our implementation is single-threaded. Our code is available on GitHub: https://github.com/RaeGBR/ZKP-Paillier-SP24. We experiment on a Linux PC with an Intel i9-12900k CPU and 64GB RAM. Our results are given in the non-interactive version.

TABLE 3. Notations and Definitions.

Notation	Definition
κ	security parameter
U, $\mathcal{S}_b$	slot size, batch size
$\mathcal{N}_p$, $\mathcal{N}_b$	number of plaintexts, number of binary messages to be proved
$\mathcal{S}_{N^2}$, $\mathcal{S}_Q$	element size in $\mathbb{Z}_{N^2}$, element size in $\mathbb{Z}_Q$
$\mathcal{N}_l$, $\mathcal{N}_m$	linear constraints number, multiplication constraints number
$\mathcal{N}_{om}$	number of multiplication operations
$\mathcal{N}_{oe}$	number of exponentiation operations
$\mathcal{T}_m$	time consumed in a single multiplication operation
$\mathcal{T}_e$	time consumed in a single exponentiation operation

Micro-benchmarks. We perform a series of micro-benchmarks to quantify the cost of each fundamental operation or element size involved in our protocol, including the time required for one exponentiation or multiplication operation and the element size in $\mathbb{Z}_{N^2}$ and $\mathbb{Z}_Q$ respectively. The corresponding data is reported in TABLE 5.[8]

TABLE 4. Parameters Used in Our Experiment. $\mathcal{N}_b/msg$ denotes the number of bits to be proved in one plaintext.

Parameter	$\|N\|$	U	$\mathcal{N}_b/msg$	$\mathcal{S}_b$	κ
$p1$	1024	32 (bits)	32	15	80
$p2$	2048	32 (bits)	64	15	128

TABLE 5. Micro-benchmarks. *The following data is the median by running* 1000 *times of corresponding operations.*

$\|N\|$	$\mathcal{T}_e$ (ms)	$\mathcal{T}_m$ (ns)	$\mathcal{S}_{N^2}$ (bits)	$\mathcal{S}_Q$ (bits)
1024	2.02	1253.54	2048	2056
2048	13.64	3770.00	4096	4112

7.1. Communication Efficiency

To demonstrate communication complexity, we test and plot total and amortized proof size in Fig. 3. Total proof size includes all the communication transmissions from the prover to the verifier, while the amortized size is computed by dividing the total size by the number of proved bits.

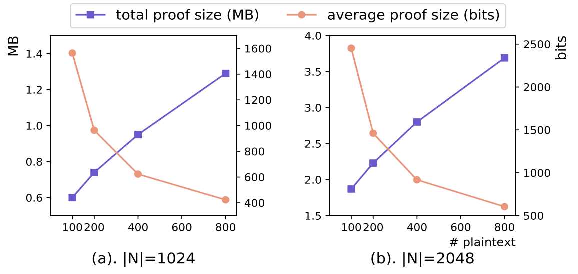

Figure 3. Total and Amortized Proof Size.

The amortized cost will decrease as the number of plaintexts increases since the total proof size is not linear to the number of proved bits. For example, when the number of proved bits increases from 6.4K to 51.2K within $|N| = 2048$ setting, the total proof size expands from 1.87 MB to 3.69 MB. Consequently, this decreases the amortized proof size from 2453 bits to 605 bits. To show the effect of batching, more data is reported in Subsection 7.4.1.

7.2. Computation Efficiency

End-to-end Performance. We develop an end-to-end prototype for two-party aggregation, which leverages the voter analysis scenario discussed in Section 1 as context. Its performance is reported in TABLE 6. The results are

8. Parameters, such as slot size U, can be adjusted to suit specific requirements. please refer to Appendix C for more discussions.

given with $\mathcal{N}_p = 800$ under both $p1$ and $p2$ parameter sets. For ease of clarity, we retain the notations introduced in Section 1. The whole protocol comprises 3 phases: (1) $\mathcal{P}_1$ structures and encrypts messages for generating a proof π; (2) $\mathcal{P}_2$ verifies π and computes $\bar{C} = \sum_{i=1}^{\mathcal{N}_p} c_i^{W_i}$; (3) $\mathcal{P}_1$ decrypts $\bar{C}$ and parses the result to obtain $\{S_j\}$.

TABLE 6. End-to-end Performance Evaluation with $\mathcal{N}_p = 800$.

Parameter	Phase 1 (s)	Phase 2 (s)	Phase 3 (ns)	Proof (MB)
$p1$	2689.36	30.00	5955	1.29
$p2$	36785.80	194.07	9896	3.69

The first three columns denote the time cost of each stage, while the last represents the proof size. When aggregating 25.6K bits with $|N| = 1024$, the protocol completes in 0.76 hours, requiring a total communication of 1.49 MB (1.29 MB for proof + 0.20 MB for ciphertexts $\{c_i\}$). For aggregating 51.2K bits under $|N| = 2048$, the required time is 10.27 hours with a total communication cost of 4.08 MB.

Examine the Workload of the Prover. As TABLE 6 indicates that the first phase requires the most computation effort, we further examine $\mathcal{P}_1$'s (i.e., the prover's) workload in this phase, grouping the process into 4 stages: (1) Encryption stage. The prover does Paillier encryption for all messages $\{m_i\}$, $\{R_j\}$, and $\{c_t^*\}$. (2) Circuit creation and value assign stage. It generates all multiplication and linear constraints specified in $\mathcal{C}_{\mathcal{R}^*}$, and assigns values to the circuit wires according to the constraints. (3) Commitment stage. This is an inner process in [46], where the prover mainly does Pedersen commitment to circuit wires. (4) Remaining stage. This internal process in [46], involves the prover computing the remaining elements for proof.

TABLE 7. Prover's Time Consumption in Each Stage

Parameter	Stage 1 (s)	Stage 2 (s)	Stage 3 (s)	Stage 4 (s)
$p1$	0.84	18.69	2570.09	99.74
$p2$	6.16	51.83	35857.20	870.63

TABLE 7 illustrates the prover's time consumption across each stage of phase 1, given $\mathcal{N}_p = 800$ under both $p1$ and $p2$ parameter settings. The commitment stage (stage 3) is the most time-consuming part. For $|N| = 1024$ and $|N| = 2048$, this stage constitutes 95.57% and 97.48% of the total proof time, respectively. Potential optimization strategies for this stage are provided in Appendix C.

7.3. Proof and Verification Cost Estimation

We also introduce methods for estimating the proof and verification costs of our main protocol. The results (TABLE 10) demonstrates that our estimation aligns well with the real data. For proving 2^{26} bits inserted into millions (i.e., 2^{20}) of 2048-bit Paillier plaintexts, our main protocol requires about 0.3 s and 2.01 ms for the prover and verifier, respectively. Furthermore, the amortized proof size is reduced to 18.85 bits. More details are available in Appendix B.

7.4. Comparison

For further illustration, we compare our schemes with potential solutions employing the state-of-the-art tools.

7.4.1. Compare ZKAoK^* with An OR-Proof [53]- Examining the Value of Packing. To evaluate the utility of packing used in our approach, we compare our main protocol with a standard OR-proof [53], where the prover uses more (unpacked) Paillier ciphertexts and proves that each plaintext encrypts a Boolean value. Furthermore, as our approach supports batch proving and verification, we report more data on varying numbers of plaintexts ($\mathcal{N}_p$), ranging from small (single or dozens) to large (hundreds), to facilitate batch size selection in practice.

The comparison result is reported in TABLE 8. In the OR-proof, the cost for each bit is constant, while in our case, the amortized cost decreases when proving more plaintexts (bits) simultaneously. In both the $|N| = 1024$ and $|N| = 2048$ settings, our method yields a smaller amortized proof size than the OR-proof when more than 20 plaintexts are being proven. In the $|N| = 2048$ setting, when over 100 plaintexts are being proven, both our proof size and verification time outperform using an OR-proof.

Our method becomes significantly more efficient as the number of plaintexts increases. When proving between 25.6K and 51.2K bits under $|N| = 2048$ setting, our amortized proof size is around 17.8x - 27x smaller. This gap will further expand to 133x when proving 10 thousand messages. As a result, our amortized bandwidth cost (proof+encryption) is only 982-670 bits, which is 20x - 30x lower than that of a standard OR-proof.

Indeed, when proving a single or a small number of messages (i.e., 10-20), our cost could remain relatively high. This is due to the need to construct auxiliary ciphertexts, c_j', to ensure soundness. Therefore, ZKAoK^* is better suited when the number of plaintexts is "larger", typically on the scale of hundreds. Conversely, for proving a single bit or several bits, the naive solution is more appropriate. Nonetheless, we underscore that this requirement is compatible with our motivating scenarios presented in Section 1, where each election poll gathers at least hundreds to thousands of messages daily. As such, our approach is ideally suited for these large-scale scenarios.

While our proof time may be longer than an OR-proof, this does not compromise the practicality of our approach in real-world applications. For instance, in a $|N| = 2048$ setting, with 800 messages collected in a day, these could be processed in one batch overnight, taking roughly 10 hours. Alternatively, if one generates proofs simultaneously for smaller batches of 200 or 400 messages, this time will be reduced to 4.73 or 6.76 hours respectively. Furthermore, one can use parallelization to improve efficiency, for facilitating deployment requirements in practice. We give more potential optimization approaches in Appendix C.

7.4.2. Compare ZKAoK' with Paillier Range Proofs [33], [54]. There are state-of-the-art methods in [33], [54] for

TABLE 8. Compare Our Approach and a Standard OR-proof [53]. *Data under $p1$ / $p2$ parameters represent amortized values for* ZKAoK* *and averages from 1000 repetitions for OR-proof. A "–" indicates non-applicability of the attribute.*

Schemes	Para.	$\mathcal{N}_p$	$\mathcal{N}_b$	Proof Size (bits)	Prove Time (s)	Verify Time (ms)	Enc. Size (bits)
OR-proof [53]	$p1$ / $p2$	–	1	>8192 / >16384	$4.08*10^{-3}$ / 0.35	2.00 / 18.04	2048 / 4096
ZKAoK*	$p1$ / $p2$	1	32 / 64	107.69K / 187.05K	14.65 / 143.38	147.94 / 779.91	**64 / 64**
		10	320 / 640	11.27K / 19.26K	1.55 / 14.80	15.90 / 82.42	
		20	640 / 1.28K	**5931.65 / 9978.29**	0.83 / 7.94	8.56 / 43.43	
		40	1.28K / 2.56K	**3228.11 / 5298.84**	0.47 / 4.44	4.81 / 23.40	
		100	3.2K / 6.4K	**1563.53 / 2453.29**	0.25 / 2.14	2.57 / **11.42**	
		200	6.4K / 12.8K	**964.24 / 1461.24**	0.17 / 1.33	**1.80 / 7.04**	
		400	12.8K / 25.6K	**623.25 / 918.20**	0.13 / 0.95	**1.38 / 5.40**	
		800	25.6K / 51.2K	**422.57 / 605.08**	0.10 / 0.70	**1.17 / 3.79**	

range proving Paillier messages. Their implementation can be found in [82]. We compare ZKAoK′ with them for range-proving 256-bit messages under $|N| = 2048$ setting (with parameter $p2$) and plot the result in TABLE 9.

TABLE 9. Compare ZKAoK′ with [33], [54] for Range Proving 256-*bit* Paillier Plaintexts in $|N| = 2048$ Setting.

Scheme	$\mathcal{N}_p$	Proof Size/Message	Total Proof Size	Proof time/Message	Verify time/Message
[82]	10 thousand	>128.00 KB	>2.00 GB	241.43 ms	199.32 ms
Ours		0.93 KB	14.87 MB	25.51 s	146.15 ms
[82]	1 million	>128.00 KB	>128 GB	241.43 ms	199.32 ms
Ours		931.84 bit	116.48 MB	20.03 s	121.08 ms
[82]	10 million	> 128.00 KB	>2048.00 GB	241.43 ms	199.32 ms
Ours		232.12 bit	0.45 GB	19.46 s	118.48 ms

* We use 2^{14}, 2^{20} and 2^{24} to estimate 10 thousand, 1 million and 10 million.

The constant communication and computation costs per plaintext in [33], [54] result in a proof size of $4096 * 2 * 128$ bits $= 128$ KB for one plaintext, which is 256 times larger than its encryption cost of 4096 bits. Besides, it requires nearly equal time for verification and proof. Our method is competitive in proof size and verification time while providing precise range proof. Specifically, we require only 11.4KB per message when proving 200 messages simultaneously. When proving 10 million messages, their approach requires $>$ 2048 GB in total, while ours requires about 0.45 GB. Our protocol's verification time per message is approximately half of theirs, even though they optimized their code with parallelization and ours uses a single thread. (Without parallelization, they require 1.67 s and 1.25 s for proof and verification, respectively.) Moreover, while they can only guarantee $x \in [0, q)$ with input $x \in [0, \lfloor q/3 \rfloor)$, our method provides precise proof.

8. Discussions

Discussion of the trade-off between proof time and proof size. Our approach significantly reduces the proof size, with increased computational overhead on the prover. This trade-off is particularly beneficial in our target application of data aggregation, which typically involves asynchronous processes of data collection and aggregation where immediate results are not critical. In these scenarios, proof generation usually occurs at or post data collection but before the aggregation phase, which might be scheduled weeks or even months later. This scheduling makes the latency in proof generation inconsequential in the overall timeline, ensuring a smooth integration with the usual data aggregation schedules. For instance, in voter analysis, our emphasis is on statistical aggregation rather than real-time tallying. Data may be gathered pre- or post-election, followed by statistical processing with various parties. The flexibility in the timing of these statistical analyses mitigates the longer proof generation time associated with our approach.

Moreover, considering a party may collaborate with different data providers, our approach's extended proof generation time is offset by the reusability of proofs across multiple parties. This feature significantly enhances overall efficiency, since the proof generation time could be amortized over multiple collaborations. Additionally, the smaller proof size is a crucial advantage in such collaborative environments, as it reduces the burden of data transfers and storage, particularly when dealing with large-scale data sets.

Discussion of Alternative Approaches. To achieve the same homomorphic functionality for the aggregation scenario, one may choose alternative approaches such as variants of Paillier over elliptic curves and lattice-based schemes. Paillier encryption [24] was first extended into the elliptic curve setting in [83]. Nevertheless, Galbraith [84] pointed out that the schemes in [83] are not secure and presented an alternative approach. As noted by the authors in [84], this approach is mainly of theoretical interest, and the resulting scheme is much slower than comparable integer computations, since it is based on elliptic curves over large numbers. Furthermore, its security analysis is not well-studied and there are no public implementations so far. Therefore, we believe that it is not mature and practical enough to be employed in reality.

In examining RLWE-based homomorphic schemes such as CKKS [85], one of the most practical homomorphic schemes, we find that it supports homomorphic operations over real numbers but is limited by error accumulation, preventing the execution of unbounded depth circuits. Accommodating a large scalar weight W_i requires a large scaling factor, leading to significant computational overhead. Its ciphertext size, at least 2^{14} bits using parameters in [85], exceeds our encryption and proof size for each message. Besides, a scalar W_i around 2^{15} will introduce a noticeable decryption error in many of the existing RLWE-based schemes. TFHE [86], while theoretically allowing unlimited homomorphic operations, is impractical in our scenario due to the large Boolean circuit and the overhead from numerous bootstrapping operations. Thus, while lattice-based schemes offer ideal properties for homomorphic operations, they don't align well with our scenario and may incur prohibitive computation and communication costs.

Adapting existing sublinear proof systems to our scenario is feasible but impractical due to the need to generate a circuit over a prime field. For a modest $|N| = 1024$ setting, the circuit for one Paillier plaintext-ciphertext pair comprises ten million gates (i.e., 2^{24}). Applying the Groth16 proof system, [45], [87], on this circuit, the trusted setup phase requires up to 120 GB of memory. Despite having a total of 132 GB memory (32 GB RAM and 100GB swap), we are unable to generate a proof due to memory limitations. The common reference string also occupies 2.88 GB, significantly larger than the ciphertext (2048 bits) itself. In $|N| = 2048$ setting, even with 160 GB memory, circuit generation remains unfeasible. Given that the best-reported implementation in [88] only supports around 2^{27} gates, and the $|N| = 2048$ circuit will be much larger, we believe it's not feasible. Applying sublinear proofs for Boolean circuits in the context of correct Paillier encryption with $|N| = 1024$ setting will lead to a circuit requiring much more than 2^{24} gates due to the use of binary operations. While [89], [90] provide efficient proofs for 0/1, they are not directly applicable to the Paiilier system, since the resulting circuit will be much larger. Therefore, directly applying existing sublinear proofs to Paillier is also not feasible due to the significant increase in the circuit size. Moreover, we provide more discussions in Appendix C.

9. Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant Nos. 61972332, 62172412), Hong Kong Research Grants Council (Project Nos. 15211120, 17201421, R1012-21, C1029-22G), PolyU Strategic Hiring Scheme (P0046340), and Ant Group.

References

[1] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference*, ser. Crypto'12, 2012, pp. 643–662.

[2] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE symposium on security and privacy (SP)*, ser. SP'17. IEEE, 2017, pp. 19–38.

[3] M. Keller, V. Pastro, and D. Rotaru, "Overdrive: making spdz great again," in *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, ser. Eurocrypt'18. Springer, 2018, pp. 158–189.

[4] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *USENIX Annual Technical Conference (USENIX ATC)*, 2020.

[5] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *IEEE symposium on security and privacy*, 2013.

[6] C. Gentry and S. Halevi, "Compressible fhe with applications to pir," in *Theory of Cryptography Conference*. Springer, 2019, pp. 438–464.

[7] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold, "Keyword search and oblivious pseudorandom functions," in *Theory of Cryptography Conference*. Springer, 2005, pp. 303–324.

[8] A. Y. Lindell, "Efficient fully-simulatable oblivious transfer," in *Cryptographers' Track at the RSA Conference*. Springer, 2008, pp. 52–70.

[9] A. Kiayias and M. Yung, "The vector-ballot e-voting approach," in *International Conference on Financial Cryptography*, ser. FC '04.

[10] I. Damgård and M. Jurik, "A generalisation, a simplification and some applications of paillier's probabilistic public-key system," in *International workshop on public key cryptography*, ser. PKC '01. Springer, 2001, pp. 119–136.

[11] M. Jawurek and F. Kerschbaum, "Fault-tolerant privacy-preserving statistics," in *Privacy Enhancing Technologies*, ser. PETS '12.

[12] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, p. 735–746.

[13] E. Shi, T.-H. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," vol. 2, 01 2011.

[14] R. A. Popa, A. J. Blumberg, H. Balakrishnan, and F. H. Li, "Privacy and accountability for location-based aggregate statistics," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, ser. CCS '11, 2011, p. 653–666.

[15] M. Joye and B. Libert, "A scalable scheme for privacy-preserving aggregation of time-series data," in *International Conference on Financial Cryptography and Data Security*, ser. FC' 13, 2013.

[16] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17, 2017, p. 1175–1191.

[17] H. Corrigan-Gibbs and D. Boneh, "Prio: Private, robust, and scalable computation of aggregate statistics," ser. USENIX'17.

[18] T. Elahi, G. Danezis, and I. Goldberg, "Privex: Private collection of traffic statistics for anonymous communication networks," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS' 14, 2014, pp. 1068–1079.

[19] L. Melis, G. Danezis, and E. De Cristofaro, "Efficient private statistics with succinct sketches," *arXiv preprint arXiv:1508.06110*, 2015.

[20] Z. Erkin and G. Tsudik, "Private computation of spatial and temporal power consumption with smart meters," in *Applied Cryptography and Network Security*, 2012.

[21] M. Zhou, T. Wang, T. Chan, G. Fanti, and E. Shi, "Locally differentially private sparse vector aggregation," in *2022 2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 1565–1565.

[22] R. Bassily and A. Smith, "Local, private, efficient protocols for succinct histograms," in *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, ser. STOC'15, pp. 127–135.

[23] M. Abdalla, J. Gong, and H. Wee, "Functional encryption for attribute-weighted sums from k-lin," in *Annual International Cryptology Conference*, ser. Crypto' 20. Springer, 2020, pp. 685–716.

[24] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology — EUROCRYPT '99*.

[25] "Homomorphic standard. iso/textbakslash 18033-6:2019," 2019. [Online]. Available: https://www.iso.org/standard/67740.html

[26] S. Agrawal, S. Badrinarayanan, P. Mukherjee, and P. Rindal, "Game-set-match: Using mobile devices for seamless external-facing biometric matching," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS'20.

[27] T. Ge and S. B. Zdonik, “Answering aggregation queries in a secure system model,” in *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*. ACM, 2007, pp. 519–530.

[28] “Alberta senate election act,” https://www.alberta.ca/alberta-senate-election-act.aspx.

[29] “Election news of federative republic of brazil,” https://www.electionguide.org/countries/id/31/.

[30] “Election for swiss council of states 2019,” https://www.electionguide.org/elections/id/3448/.

[31] “2021 election committee subsector ordinary elections,” https://www.elections.gov.hk/ecss2021/eng/brief.html.

[32] S. Goldwasser, S. Micali, and C. Rackoff, “The knowledge complexity of interactive proof-systems,” in *STOC ’85*, 1985, pp. 291–304.

[33] Y. Lindell, “Fast secure two-party ecdsa signing,” in *Advances in Cryptology – CRYPTO 2017*, ser. Crypto ’17, 2017, pp. 613–644.

[34] Y. Lindell and A. Nof, “Fast secure multiparty ecdsa with practical distributed key generation and applications to cryptocurrency custody,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1837–1854.

[35] R. Cramer, I. Damgård, and J. B. Nielsen, “Multiparty computation from threshold homomorphic encryption,” in *Advances in Cryptology — EUROCRYPT 2001*, ser. Eurocrypt ’01, 2001, pp. 280–300.

[36] C. Hazay and Y. Lindell, “Efficient oblivious polynomial evaluation with simulation-based security,” *Cryptology ePrint Archive*, 2009.

[37] “Election administration and voting survey 2020 comprehensive report (us).” [Online]. Available: https://www.eac.gov/sites/default/files/document_library/files/2020_EAVS_Report_Final_508c.pdf

[38] “Poll of polls - polling from across europe.” [Online]. Available: https://www.politico.eu/europe-poll-of-polls

[39] “Number of early votes cast in the 2020 presidential election in the united states.” [Online]. Available: https://www.statista.com/statistics/1184422/presidential-election-number-early-votes-cast-us/

[40] “2022 general election early voting statistics.” [Online]. Available: https://rpubs.com/ElectProject/early_vote_2022

[41] J. Groth, “Linear algebra with sub-linear zero-knowledge arguments,” in *Annual International Cryptology Conference*, ser. Crypto’ 09. Springer, 2009, pp. 192–208.

[42] B. Parno, J. Howell, C. Gentry, and M. Raykova, “Pinocchio: Nearly practical verifiable computation,” ser. SP, 2013.

[43] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, “Snarks for c: Verifying program executions succinctly and in zero knowledge,” in *Annual cryptology conference*. Springer, 2013.

[44] M. Jawurek, F. Kerschbaum, and C. Orlandi, “Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently,” in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, ser. CCS’ 13, 2013, pp. 955–966.

[45] J. Groth, “On the size of pairing-based non-interactive arguments,” in *Advances in Cryptology–EUROCRYPT 2016*.

[46] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit, “Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting,” in *Advances in Cryptology – EUROCRYPT 2016*.

[47] I. Giacomelli, J. Madsen, and C. Orlandi, “Zkboo: Faster zero-knowledge for boolean circuits,” in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 1069–1083.

[48] J. Bootle, A. Cerulli, E. Ghadafi, J. Groth, M. Hajiabadi, and S. K. Jakobsen, “Linear-time zero-knowledge proofs for arithmetic circuit satisfiability,” ser. AsiaCrypt’ 17. Springer, 2017, pp. 336–365.

[49] B. Bunz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, “Bulletproofs: Short proofs for confidential transactions and more,” in *2018 IEEE Symposium on Security and Privacy (SP)*.

[50] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn, “Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings,” ser. CCS’19.

[51] M. Hoffmann, M. Klooß, and A. Rupp, “Efficient zero-knowledge arguments in the discrete log setting, revisited,” ser. CCS, 2019.

[52] J. Zhang, T. Xie, Y. Zhang, and D. Song, “Transparent polynomial delegation and its applications to zero knowledge proof,” in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020.

[53] R. Cramer, I. Damgård, and B. Schoenmakers, “Proofs of partial knowledge and simplified design of witness hiding protocols,” in *Annual International Cryptology Conference*, 1994.

[54] F. Boudot, “Efficient proofs that a committed number lies in an interval,” in *Advances in Cryptology — EUROCRYPT 2000*.

[55] C. Ordonez, “Clustering binary data streams with k-means,” in *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 2003, pp. 12–19.

[56] M. Zhang and N. Hurley, “Avoiding monotony: improving the diversity of recommendation lists,” in *Proceedings of the 2008 ACM conference on Recommender systems*, 2008, pp. 123–130.

[57] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas, “Adnostic: Privacy preserving targeted advertising,” in *Proceedings Network and Distributed System Symposium*, 2010.

[58] S. Li, N. Vlassis, J. Kawale, and Y. Fu, “Matching via dimensionality reduction for estimation of treatment effects in digital marketing campaigns.” in *IJCAI*, 2016, pp. 3768–3774.

[59] R. Li, S. Wang, and K. C.-C. Chang, “Multiple location profiling for users and relationships from social network and content,” *arXiv preprint arXiv:1208.0288*, 2012.

[60] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, “Effects of user similarity in social media,” in *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012.

[61] “Google adwords,” https://ads.google.com/home/.

[62] “Google adsense,” https://adsense.google.com.

[63] C. Wang, R. Raina, D. Fong, D. Zhou, J. Han, and G. Badros, “Learning relevance from heterogeneous social network and its application in online targeting,” in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*.

[64] J. Yan, N. Liu, G. Wang, W. Zhang, Y. Jiang, and Z. Chen, “How much can behavioral targeting help online advertising?” ser. WWW.

[65] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, “Diversified texture synthesis with feed-forward networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[66] H. Hajishirzi, W.-t. Yih, and A. Kolcz, “Adaptive near-duplicate detection via similarity learning,” in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 2010, pp. 419–426.

[67] E. Shi and K. Wu, “Non-interactive anonymous router,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2021, pp. 489–520.

[68] D. Heath and V. Kolesnikov, “One hot garbling,” in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 574–593.

[69] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua, “Explicit filterbank learning for neural image style transfer and image processing,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 7, pp. 2373–2387, 2020.

[70] P. Zhang, M. Wu, H. Dinkel, and K. Yu, “Depa: Self-supervised audio embedding for depression detection,” in *Proceedings of the 29th ACM international conference on multimedia*, 2021, pp. 135–143.

[71] B. Guo, S. Han, X. Han, H. Huang, and T. Lu, “Label confusion learning to enhance text classification models,” in *Proceedings of the AAAI conference on artificial intelligence*, 2021.

[72] J. Li, X. Lan, Y. Long, Y. Liu, X. Chen, L. Shao, and N. Zheng, "A joint label space for generalized zero-shot classification," *IEEE Transactions on Image Processing*, vol. 29, pp. 5817–5831, 2020.

[73] B. Liu, Y. Zhu, Z. Fu, G. De Melo, and A. Elgammal, "Oogan: Disentangling gan with one-hot sampling and orthogonal regularization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[74] R. Diaz and A. Marathe, "Soft labels for ordinal regression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, ser. CVPR'19, 2019, pp. 4738–4747.

[75] J. Camenisch and M. Michels, "Proving in zero-knowledge that a number is the product of two safe primes," ser. Eurocrypt' 99.

[76] C. Hazay, G. L. Mikkelsen, T. Rabin, T. Toft, and A. A. Nicolosi, "Efficient rsa key generation and threshold paillier in the two-party setting," *Journal of Cryptology*, p. 265–323, 2019.

[77] J. Boyar, K. Friedl, and C. Lund, "Practical zero-knowledge proofs: Giving hints and using deficiencies," *Journal of cryptology*, 1991.

[78] J. v. d. Graaf and R. Peralta, "A simple and secure way to show the validity of your public key," in *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 1987.

[79] A. Chan, Y. Frankel, and Y. Tsiounis, "Easy come—easy go divisible cash," in *International Conference on the Theory and Applications of Cryptographic Techniques*, ser. Eurocrypt' 98. Springer, 1998, pp. 561–575.

[80] E. Fujisaki and T. Okamoto, "Statistical zero knowledge protocols to prove modular polynomial relations," in *Annual International Cryptology Conference*, ser. Crypto' 97. Springer, 1997.

[81] V. Shoup, "Ntl: a library for doing number theory," 1996.

[82] "Range proof for paillier," https://github.com/ZenGo-X/zk-paillier/blob/master/src/zkproofs/range_proof_ni.rs, 2018.

[83] P. Paillier, "Trapdooring discrete logarithms on elliptic curves over rings," in *Advances in Cryptology—ASIACRYPT 2000: 6th International Conference on the Theory and Application of Cryptology and Information Security*.

[84] S. D. Galbraith, "Elliptic curve paillier schemes," *Journal of Cryptology*, vol. 15, pp. 129–138, 2002.

[85] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," ser. ASIACRYPT'17. Springer.

[86] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Tfhe: fast fully homomorphic encryption over the torus," *Journal of Cryptology*, 2020.

[87] "Arkworks - an ecosystem for developing and programming with zksnarks," https://github.com/arkworks-rs.

[88] B. Chen, B. Bünz, D. Boneh, and Z. Zhang, "Hyperplonk: Plonk with linear-time prover and high-degree custom gates," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2023, pp. 499–530.

[89] I. Cascudo and E. Giunta, "On interactive oracle proofs for boolean r1cs statements," in *International Conference on Financial Cryptography and Data Security*. Springer, 2022, pp. 230–247.

[90] Y. Gvili, S. Scheffler, and M. Varia, "Booligero: Improved sublinear zero knowledge proofs for boolean circuits," in *Financial Cryptography and Data Security: 25th International Conference, FC 2021.*

[91] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology — CRYPTO '91.*

[92] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, "Efficient garbling from a fixed-key blockcipher," in *2013 IEEE Symposium on Security and Privacy.*

[93] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole: Reducing data transfer in garbled circuits using half gates," in *Advances in Cryptology-EUROCRYPT 2015.*

[94] P. Rindal and M. Rosulek, "Faster malicious 2-party secure computation with online/offline dual execution," in *25th USENIX Security Symposium*, 2016.

[95] X. Wang, S. Ranellucci, and J. Katz, "Authenticated garbling and efficient maliciously secure two-party computation," ser. CCS'17.

[96] N. Dottling, S. Ghosh, J. B. Nielsen, T. Nilges, and R. Trifiletti, "Tinyole: Efficient actively secure two-party computation from oblivious linear function evaluation," ser. CCS'17.

Appendix A. Additional Preliminaries

In this section, we present additional definitions of the underlying preliminaries. Roughly speaking, the decisional composite residuosity assumption says that given an RSA modulus N with an element $z \in \mathbb{Z}^*_{N^2}$, the adversary cannot distinguish (except with negligible probability) whether z is an N-th residue.

Definition A.1. (Decisional Composite Residuosity (DCR) Assumption). *This assumption holds relative to the key generation algorithm* $\mathsf{PL.Gen}$ *in Paillier system, if for all non-uniform PPT adversaries* $\mathcal{A}$*, the following holds,*

$$\left|\Pr\left[\mathcal{A}(N,z_1)=1\,\middle|\,(N,p,q)\leftarrow \mathsf{PL.Gen}(1^\lambda), r_1\leftarrow \mathbb{Z}^*_N, z_1=r_1^N \mod N^2\right]\right.$$
$$\left.-\Pr\left[\mathcal{A}(N,z_2)=1\,\middle|\,(N,p,q)\leftarrow \mathsf{PL.Gen}(1^\lambda), r_2\leftarrow \mathbb{Z}^*_{N^2}, z_2=r_2\right]\right|\approx 0.$$

Paillier encryption, based on the DCR assumption, contains 3 algorithms, $\mathsf{PL} = (\mathsf{PL.Gen}, \mathsf{PL.Enc}, \mathsf{PL.Dec})$. Since there are many variants, we use a simpler version in our paper where p and q are set as equal length and $N = pq$ with setting base as $(1 + N)$. Readers can refer to [24] for its formal definition and security proof.

A Pedersen commitment scheme over composite order group contains 2 polynomial algorithms, $\mathsf{PDC} = (\mathsf{PDC.Gen}, \mathsf{PDC.Com})$. With input security parameter λ, $\mathsf{PDC.Gen}$ outputs the commitment key, $ck = (\mathbb{G}, g, h)$, where $\mathbb{G}$ is a cyclic group of composite order satisfying Def.3.1, g denotes its generator and h is a randomly chosen element. With input ck and message m, $\mathsf{PDC.Com}$ outputs its commitment as, $ct = g^m \cdot h^r \mod Q$, s.t., $Q = f \cdot N^2 + 1$ is a prime where f is a random small integer. In our protocol, we use a variant of the above form. The commitment key is set as $ck = (\mathbb{G}, g, g_1, g_2, \ldots, g_n)$ and a commitment of a series of messages, $(m_1, \ldots, m_n)$, is, $ct = g^r \cdot \Pi_{i=1}^n g_i^{m_i}$. We refer [91] for more details.

A.1. Zero-knowledge Argument of Knowledge

Our protocol is a zero-knowledge argument of knowledge (ZKAoK) [9], which should satisfy perfect completeness, statistical witness-extended emulation, and statistical special honest-verifier zero-knowledge. We use witness emulation to define knowledge soundness, as used in [46]. Informally, it states that there exists an emulator that can produce an argument with identical distribution, along with a witness, whenever an adversary can produce an argument satisfying

9. Our protocol is actually an "argument" and we do not specifically distinguish two terms, "argument" and "proof", in our paper.

the verifier with some probability. For zero-knowledge, it means that given the verifier's challenge values in advance, it is possible to efficiently simulate the entire argument without knowing the witness. Due to space limitations, we refer [46] for a more detailed and formal definition.

Appendix B. Estimation of Proof and Verification Cost

In this section, we present detailed estimation approaches for communication and computation costs, towards our main protocol. We denote the number of multiplication and linear constraints as $\mathcal{N}_m$ and $\mathcal{N}_l$ respectively, where $\mathcal{N}_m$ can be computed as,

$$\mathcal{N}_m = \frac{|N|}{U} \cdot \mathcal{N}_p + (\mathcal{N}_p + \kappa + \lceil \frac{|N|}{\mathcal{S}_b} \rceil) \cdot (\lceil \log N \rceil + h(N) - 1)$$

We estimate the total communication cost as,

$$\mathcal{S}_{N^2} \cdot O(\mathcal{N}_p/U + \sqrt{\mathcal{N}_p} + \kappa) + \mathcal{S}_Q \cdot O(\sqrt{\mathcal{N}_m} + \sqrt{\sqrt{\mathcal{N}_m}}).$$

We use commitment time, the most significant component, to represent the proof time, as analyzed in Section 7.2. We denote the quantities of exponentiation and multiplication operations as $\mathcal{N}_{oe}$ and $\mathcal{N}_{om}$, which can be calculated as,

$$\begin{aligned} \mathcal{N}_{oe} &= (|N|/4 + 1) * (3\sqrt{\mathcal{N}_m} + 1)/2 \\ \mathcal{N}_{om} &= |N| * (3\sqrt{\mathcal{N}_m} + 1) * (\sqrt{\mathcal{N}_m} + 512)/4. \end{aligned}$$

We use $\mathcal{T}^{est.}_{prf} = \mathcal{N}_{oe} * \mathcal{T}_e + \mathcal{N}_{om} * \mathcal{T}_m$ to estimate the total proof time. For verification time, it can be estimated as,

$$\mathcal{T}^{est.}_{vrf} = O(\mathcal{N}_l + \mathcal{N}_m) * \mathcal{T}_m + O(2\sqrt{\mathcal{N}_m} + 1) * \mathcal{T}_e.$$

TABLE 10. COMPARISON OF REAL AND ESTIMATED AMORTIZED DATA.

Par.	$\mathcal{N}_p$	Proof Size		Prove Time		Verify Time	
		est. (bits)	est. (bits)	real (s)	est. (s)	real (ms)	est. (ms)
$p1$	200	964.242	969.35	0.17	0.20	1.80	1.88
	400	623.25	626.68	0.13	0.15	1.38	1.42
$p2$	200	1461.24	1464.77	1.33	1.57	7.04	8.20
	400	918.20	920.93	0.95	1.08	5.40	5.78

We compare our estimated amortized data towards proof size, proof and verification time with the real data in TABLE 10. The amortized cost is calculated by dividing the total cost by the number of proved bits (i.e., binary records). The result demonstrates that our estimation aligns closely with the real data. For proving 2^{26} bits inserted into millions of 2048-bit Paillier plaintexts, it will cost about 0.3 s and 2.01 ms for proving and verifying one bit respectively, while the amortized proof size will be 18.85 bits only.

Appendix C. Potential Optimizations and Discussions

Optimization. Much of the workload in the commitment phase can be pre-computed, allowing for a division of our protocol into online-offline phases. Here we have a closer look at the involved multiplication constraints. Most of them have the Paillier encryption form as, $c = (1 + N)^m \cdot r^N$ (while others have the form, $b * (b - 1) = 0$). We need to construct 1 linear constraint and $(\lceil \log N \rceil + h(N) - 1)$ multiplication constraints to represent $1 + m \cdot N$ and r^N respectively. As these multiplication constraints are only related to randomness, not witness, one could pre-commit them before running our protocol. The number of multiplication constraints that can be pre-computed is,

$$(\mathcal{N}_p + \kappa + \lceil \frac{\mathcal{N}_p}{\mathcal{S}_b} \rceil) \cdot (\lceil \log N \rceil + h(N) - 1).$$

That is, over 98% of commitments can be pre-computed. Besides, as our implementation uses single-thread, we can further optimize it by applying parallelization.

Discussion of Proving Other Plaintext Relations Using Our Approach. Here we discuss how to additionally prove other Paillier plaintext relations using our approach. For example, to prove the relation $\mathcal{R}_{\mathsf{mul}}$ in [10], one can use our method by adding one constraint, $m_3 = m_1 * m_2$, into the system. One thing that needs to be argued is that our constraint is under modulo N^2 where the original relation is under modulo N. We now give the following observation.

If we have $m_3 = m_1 \cdot m_2 \mod N^2$, then we can rewrite it as $m_3 = m_1 \cdot m_2 + kN^2$ for some $k \in \mathbb{Z}$. Then we have, $m_3 = m_1 \cdot m_2 + kN \cdot N = m_1 \cdot m_2 \mod N$. We use c_1, c_2, c_3 to denote their corresponding ciphertexts. Even if we put $m_1 > N$, say, $m_1 = m_1' + k_1 N$ (resp. $m_2 = m_2' + k_2 N$ and $m_3 = m_3' + k_3 N$) into the encryption, the actual value being encrypted is still m_1' (resp. m_2' and m_3'). So if $m_3 = m_1 \cdot m_2 \mod N^2$ holds with having $\mathsf{PL.Enc}(m_1; r_1)$, $\mathsf{PL.Enc}(m_2; r_2)$ and $\mathsf{PL.Enc}(m_3; r_3)$, we can only get m_1', m_2' and m_3' after decryption. Therefore we still have $m_3' = m_1' \cdot m_2' \mod N$.

Discussion of Using a Generic Two-party Computation Protocol. While generic two-party computation protocols can potentially provide similar functionality, current efficient advances [92], [93] can only achieve semi-honest security. To achieve malicious security, one can additionally use a generic ZKP. However, this will bring in large overhead and break its efficiency advantage. Although one might consider using adaptive-secure two-party computation protocols, existing works [94], [95] fall short in terms of round and communication efficiency. Customized maliciously secure OT/OLE schemes [96] could be an alternative but these require greater computation and communication resources. We emphasize our non-interactive approach wherein $\mathcal{P}_1$ can reuse proofs across collaborations. In contrast, a two-party computation protocol necessitates $\mathcal{P}_1$ to rerun the scheme with each collaborator, increasing the complexity.

Discussion of the choice of slot size and the number of slots. For ease of concreteness, we have fixed these parameters, although they can be adjusted according to the problem size and its requirements. For instance, the slot size U can be decreased or increased, provided that it meets the condition, $\mathcal{N}_p * \max\{W_i\} < 2^U$. Similarly, the number of slots ($\mathcal{N}_b/msg$) in each plaintext can also be adjusted but should satisfy $\mathcal{S}_b * \mathcal{N}_b/msg < |N|/2$. This condition is derived from Lemma 6.2 and is necessary for constructing auxiliary message m_t^*.

Appendix D. Deferred Security Analysis

D.1. Lemma D.1 and Its Proof

Lemma D.1. *Assume N is the RSA modulo, says $N = pq$ where p, q are primes. There exists 4 roots of b_i for the following equation,*

$$b_i(b_i - 1) = 0 \mod N^2.$$

Specifically, the roots are 0, 1, $X = q^2 \cdot [(q^2)^{-1} \mod p^2]$ and $Y = p^2 \cdot [(p^2)^{-1} \mod q^2]$.

Proof. (of Lemma D.1) We start by showing that $x(x-1) = 0 \mod p^2$ (resp. $\mod q^2$) has exactly two roots, 0 mod p^2 and $1 \mod p^2$ (resp. $0 \mod q^2$ and $1 \mod q^2$). It is obvious that $x(x-1) = 0 \mod p$ has two roots, 0 mod p and $1 \mod p$. Thus, for equaiton $x(x-1) = 0$ mod p^2, the root must be in the form, $kp + 0 \mod p^2$ or $kp + 1 \mod p^2$, for some number k. We further have that $k = 0 \mod p$. Finally, $x(x-1) = 0 \mod p^2$ has exactly two roots, $0 \mod p^2$ and $1 \mod p^2$. Similarly, the roots for $x(x-1) = 0 \mod q^2$ are, $0 \mod q^2$ and $1 \mod q^2$.

According to the Chinese Reminder Theorem, and illustrated in TABLE 11, the roots for $b_i(b_i - 1) = 0 \mod N^2$ are $0, 1, X$ and Y, where $X = q^2 \cdot [(q^2)^{-1} \mod p^2]$ and $Y = p^2 \cdot [(p^2)^{-1} \mod q^2]$.

□

TABLE 11. POSSIBLE VALUES OF b_i

Value of b_i	0	1	X	Y
Value of $b_i \mod p^2$	0	1	1	0
Value of $b_i \mod q^2$	0	1	0	1

D.2. Proof of Theorem 6.1

Since the proof of Theorem 6.1 requires the condition $\mathcal{N}_b + \max\{R'_j\} < \min\{X, Y\} < N^2 - \mathcal{N}_b - \max\{R'_j\}$ to apply Lemma 6.1, we first show that this condition is fulfilled in our application scenario. Define s_d and λ as the statistical and security parameters.

Let $M = \mathcal{N}_b + \max\{R'_j\}$. We show that if $\min\{p^2, q^2\} > M$ holds, the condition $M < \min\{X, Y\} < N^2 - M$ is satisfied. First, recall that w.l.o.g. $X = q^2 \cdot [(q^2)^{-1} \mod p^2]$ and $Y = p^2 \cdot [(p^2)^{-1} \mod q^2]$. It follows that $\max\{X\} = q^2 \cdot (p^2 - 1) = N^2 - q^2$ and $\max\{Y\} = N^2 - p^2$. Similarly, we have $\min\{X\} = q^2$ and $\min\{Y\} = p^2$. Given $\min\{p^2, q^2\} > M$, it leads to $\min\{X, Y\} = \min\{p^2, q^2\} > M$, thereby satisfying the first inequality. For the second inequality, we have $\min\{X, Y\} < \max\{X, Y\} = \max\{N^2 - p^2, N^2 - q^2\} = N^2 - \min\{p^2, q^2\} < N^2 - M$.

In our application scenario with $\mathcal{N}_p \leq 2^{20}$, we have $\mathcal{N}_b = |N| * \mathcal{N}_p/32$, where $N = \mathsf{PL.Gen}(1^\lambda)$. We set $\max\{R'_j\} = 2^{s_d} * \mathcal{N}_b$. For $\lambda = 128$, $|N| = 2048$, we can set $s_d = 255$. Then $\mathcal{N}_b \leq 2^{26}$ and $M < 2^{400}$. Assuming p and q are of equal length, then $\min\{p^2, q^2\} > M$ is readily met. To ensure this is true, we can require the private key owner to prove in zero-knowledge that N is correctly generated (from 2 equal-length primes).

Proof. (of Theorem 6.1) Define security and statistical parameters as λ and s_d. We have $\kappa = poly(\lambda)$ and $\mathcal{N}_b = |N| * \mathcal{N}_p/32$, where $N = \mathsf{PL.Gen}(1^\lambda)$. We set $\max\{R'_j\} = 2^{s_d} * \mathcal{N}_b$. Perfect completeness is derived from the completeness property of the underlying protocol and straightforward inspection, as all valid witnesses satisfy the constraints specified in $\mathcal{C}_{\mathcal{R}^*}$. Special honest verifier zero-knowledge comes from the security of Paillier encryption and the underlying zero-knowledge argument system. Specifically, (an honestly generated) S'_j always satisfies, $S'_j = \sum_{i=1}^{\mathcal{N}_p} \sum_{s=1}^{|N|/32} l^{(i)}_{j,32(s-1)} b^{(i)}_{32(s-1)} \leq \mathcal{N}_b$ (where we fix the slot size as 32-bit). Consequently, the statistical distance between the distributions of R'_j and $L'_j = S'_j + R'_j$ is bounded by $\frac{1}{2^{s_d}}$. With choosing an appropriate value of s_d, the two distributions will be statistically indistinguishable ensuring that L'_j will not leak information about S'_j.

For witness extended emulation, it is sufficient to prove that $\mathcal{R}^*$ is equivalent to $\mathcal{C}_{\mathcal{R}^*}$, since the underlying protocol already satisfies this property. It is easy to check that $\mathcal{R}^*$ implies $\mathcal{C}_{\mathcal{R}^*}$. To prove that $\mathcal{C}_{\mathcal{R}^*}$ implies $\mathcal{R}^*$, we show that if $\mathcal{C}_{\mathcal{R}^*}$ holds, then each $b^{(i)}_{32(s-1)}$ is from $\{0, 1\}$ with all but negligible probability. Using Lemma 6.2, we show that the Paillier encryption, with input message satisfying $|m| < \min\{|p|, |q|\}$, can be treated as a commitment scheme with the binding property. Consequently, all $\{b^{(i)}_{32(s-1)}\}$'s (which are decomposition bits of $\{m_t^*\}$'s) are fixed, as m_t^* satisfies this length requirement. By applying Lemma E.1, we can have that $b^{(i)}_{32(s-1)} \in \{0, 1, X, Y\}$. Therefore, by deploying Lemma 6.1, we can prove that if every L'_j satisfies $L'_j \leq M$, where $M = \mathcal{N}_b + \max\{R'_j\}$ and $j \in [1, \kappa]$, then all $\{b^{(i)}_{32(s-1)}\}$'s, where $i \in [1, \mathcal{N}_p]$ and $s \in [1, |N|/32]$, are from 0 or 1 except with probability $\frac{1}{2^\kappa} = \frac{1}{2^{poly(\lambda)}}$ (i.e., soundness error), which is negligible in λ. We can, therefore, prove that our main protocol satisfies witness extended emulation by calling the extractor in the underlying protocol to extract a valid witness.

In particular, when we set $\lambda = 128$, $|N| = 2048$, $s_d = 255$, and $\mathcal{N}_p \leq 2^{20}$ in our application scenario, we have $\mathcal{N}_b \leq 2^{26}$ and $\max\{R'_j\} = 2^{281}$. Therefore, the statistical distance between the two distributions of R'_j and L'_j is bounded by $\frac{1}{2^{255}}$, which is negligible. As a result, the zero-knowledge property can be satisfied. The soundness error is bounded by $\frac{1}{2^\kappa} = \frac{1}{2^{poly(128)}}$, which is also negligible. This implies that our protocol satisfies perfect completeness, statistical special honest verifier zero-knowledge, and statistical witness-extended emulation in our considered scenario, which completes our proof.

□