An Improvement of Multi-Exponentiation with Encrypted Bases Argument: Smaller and Faster

Yi Liu[1,2], Qi Wang[1], and Siu-Ming Yiu[2]

[1] Department of Computer Science and Engineering,
Southern University of Science and Technology
liuy7@mail.sustech.edu.cn
wangqi@sustech.edu.cn
[2] Department of Computer Science,
The University of Hong Kong
smyiu@cs.hku.hk

Abstract. A cryptographic framework, called encryption switching protocol (ESP), has been proposed recently, which enables ciphertexts encrypted under *different* schemes to be converted to the same scheme without revealing the plaintexts. This solves a major issue in privacy-preserving applications, in which users can now encrypt their data under different schemes and still be able to process their encrypted data together. In this paper, we propose an improvement to ESP. In particular, we consider the multi-exponentiation with encrypted bases argument (MEB) protocol, which is not only the essential component and efficiency bottleneck of ESP, but also has tremendous potential in many applications and can be used to speed up many intricate cryptographic protocols, such as proof of knowledge of a double logarithm. Based on our analysis and experiments, our proposed MEB protocol can reduce the communication cost by 36% when compared to the original protocol and reduce the computation cost of the verifier by $20\% - 47\%$ depending on the settings of experimental parameters. This is particularly useful for verifiers with weak computing power. We also provide a formal security proof to confirm the security of the improved MEB protocol.

Keywords: Encryption switching protocols · Paillier encryption · Twin-ciphertext proof · Zero-knowledge.

1 Background

Nowadays, data has been widely regarded as a kind of valuable resource. Many solutions have been proposed to preserve the privacy of data during its usage, such as secure multi-party computation (MPC) [9, 16, 17] and fully homomorphic encryption (FHE) [8]. However, efficiency is still a problem in most cases.

In 2016, Couteau, Peters, and Pointcheval [6] proposed a cryptographic framework named *encryption switching protocol* (ESP) (for its extension, see [4]), and it was shown that ESP has great potential to achieve many privacy-preserving goals efficiently. In ESP, two parties secretly share the private keys of an additively homomorphic encryption scheme and a multiplicatively homomorphic encryption scheme, such that the two parties can individually encrypt messages, but should cooperate to perform threshold decryption in order to decrypt a ciphertext. Two parties can also work interactively to switch one underlying encryption scheme of ciphertext to the other. In summary, ESP allows both parties to perform both additions and multiplications on encrypted values to evaluate pre-deterministic circuits securely. It was shown that ESP could be instantiated for generic two-party computation (2PC) protocol [6], and thus ESP is powerful to cover many MPC tasks (see examples in [5]).

To guarantee the correct execution of the encryption switching procedure of ESP in the presence of malicious parties, the authors of [6] introduced a new cryptographic primitive, called *twin-ciphertext proof* (TCP). We call a ciphertext pair $(C_+, C_\times)$ *twin-ciphertext* if the additively homomorphic encryption ciphertext (or commitment) C_+ is encrypting (or committing) the same message as that encrypted by the ciphertext $C_\times$ of multiplicatively homomorphic encryption. Given the values and randomness of both a random twin-ciphertext pair $(C_+, C_\times)$ and another pair (C_1, C_2), TCP allows a prover to *efficiently* prove to the verifier that (C_1, C_2) is a twin-ciphertext pair without revealing the encrypted value and its randomness of (C_1, C_2). The main idea of TCP is to open the colinear relation between the random twin-ciphertext pair and the pair for the proof, and thus the random twin-ciphertext pair $(C_+, C_\times)$ is consumed during this approach. Therefore, to speed up the online TCP process, the prover can generate a pool of random twin-ciphertext pairs at the beginning and consume the pairs one by one during the process. We note that this approach is similar to the Beaver triples technique [2].

Since a costly *cut-and-choose* procedure has to be involved in the generation of random twin-ciphertexts, the authors of [6] mentioned that it is possible to batch the executions of TCP. More precisely, by consuming one random twin-ciphertext pair, we are able to simultaneously prove that some given pairs are all twin-ciphertext pairs. This technique can be used to batch the generation of random twin-ciphertext pairs or conduct TCP for many pairs simultaneously. This idea was used in their protocol, called *multi-exponentiation with encrypted bases argument* (MEB). The MEB protocol is designed for additively homomorphic encryption (or commitment) schemes, and it acts as the underlying basis to batch the executions of TCP. Informally, given parameters $(\lambda)_{i=1,\ldots,\ell}$ and additively homomorphic encryption ciphertexts $((c_i)_{i=1,\ldots,\ell}, C)$, the MEB protocol allows a prover to prove the knowledge of encrypted values $((m_i)_{i=1,\ldots,\ell}, M)$ and randomness inside $((c_i)_{i=1,\ldots,\ell}, C)$, respectively, and the fact that the encrypted value M inside C satisfies $M = \prod_{i=1}^{\ell} m_i^{\lambda_i}$, in a zero-knowledge manner. The basic idea of batch TCP is to batch the two ciphertexts of all pairs separately in a multi-exponentiation form and execute a TCP for the pair of batched ciphertexts. The MEB protocol is indeed the bottleneck of efficiency for the execution of batch TCP.

TCP, as the direct application of the MEB protocol, is not only the underlying protocol of ESP but also of independent interest. Many commonly used MPC protocols that are expensive in traditional scenarios become very cheap when TCP is involved [6], *e.g.*, proof of knowledge of exponential relation of committed values (for both plain/committed exponent), proof of knowledge of a double logarithm, proof of committed prime, etc. Hence, the improvements of MEB can further enhance the performance of these protocols.

Moreover, MEB can individually play as an underlying protocol of some applications that are typical in commercial and medical areas. For instance, users may wish to evaluate a public function f on an encrypted dataset $(d_i)_{i=1,\ldots,m}$ provided by a data holder, where f is of the form $f = \prod_j (\sum_i a_i d_i)^{\lambda_j}$ with public constants $\{a_1, a_2, \ldots\}$ and $\{\lambda_1, \lambda_2, \ldots\}$. The MEB protocol can thus be used for the data holder to prove the correctness of the encrypted evaluation results without revealing other information of the dataset. Compared with FHE-based solutions, this approach provides a relatively smaller encrypted dataset and is much more efficient for functions with higher depths of multiplication.

In this paper, we provide an improved MEB protocol, in the sense that our protocol is more efficient for both the prover and the verifier, and has lower communication cost than the original MEB protocol in [6]. Our MEB protocol is the same as the original protocol, and it is also a public-coin special honest-verifier zero-knowledge (SHVZK) argument of knowledge (see more information in Section 2). In general, the argument size of our protocol is roughly 36% smaller than that of the original protocol. Meanwhile, our protocol reduces the computation cost of the verifier by $20\% - 47\%$ depending on different experimental parameters. The basic idea of our protocol is that we further decompose statements into several conditions and batch them into one proof of a specific relationship to obtain a compact and more efficient protocol (see more details in Section 3 and Section 4.1).

In the following, we summarize the main contributions of this paper.

1. We provide an improvement of the MEB protocol in both argument size and efficiency. To be comparable with the original MEB protocol of [6], we present the construction of our MEB protocol based on Paillier encryption [14]. We remark that MEB protocol for other additively homomorphic schemes, such as Pedersen commitment scheme [15], can be constructed in a similar approach.
2. We provide *proof-of-concept* implementations for both our MEB protocol and the original MEB protocol. We compare the two protocols from the perspectives of theoretical analysis and experiments to verify the improvement of our MEB protocol.

The rest of this paper is organized as follows. In Section 2, we introduce some necessary background knowledge. We provide the description of our MEB protocol and the corresponding subprotocols in Section 3 and Section 4, respectively. Comparisons between our protocol and the original protocol are presented in Section 5 from both theoretical and experimental aspects. We conclude this paper with future work in Section 6.

2 Preliminaries

In this paper, we mainly focus on constructing a public-coin SHVZK argument of knowledge and prove its security under standard security definitions (see [12, 13] for more information). Note that such a protocol can be compiled to be secure against malicious verifiers with low overhead by many techniques, such as using an equivocal commitment scheme [3] and adopting the Fiat–Shamir heuristic [7].

2.1 Notation

We write $x \leftarrow_{\$} S$ for uniformly sampling x from a set S. We use bold letters to represent vectors, *e.g.*, $\boldsymbol{m} = (m_1, \ldots, m_\ell)$ is a vector with ℓ entries. The notation $\boldsymbol{ab}$ denotes the entry-wise product of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, *i.e.*, $\boldsymbol{ab} = (a_1b_1, \ldots, a_\ell b_\ell)$, and the notation $r\boldsymbol{a}$ denotes scalar multiplications, *i.e.*, $r\boldsymbol{a} = (ra_1, \ldots, ra_\ell)$. The notation $||n||$ is used to represent the length of the bit-representation of a given variable n, and the notation $|S|$ denotes the size of a given set S.

We say a function f in variable μ mapping natural numbers to $[0, 1]$ is *negligible* if $f(\mu) = \mathcal{O}(\mu^{-c})$ for every constant $c > 0$. We say that $1-f$ is *overwhelming* if f is negligible. In our protocols, we will give a security parameter μ written in unary as input to all parties.

In the following descriptions of protocols, P denotes the prover, and V denotes the verifier.

2.2 Paillier Encryption

Paillier encryption scheme is a public-key additively homomorphic encryption scheme with semantic security [10] based on the *Decisional Composite Residuosity assumption*. The public key of Paillier encryption scheme is a strong RSA modulus $n = pq$, where p and q are safe primes with the same length. We denote the Paillier encryption algorithm as Enc, and thus encrypting a value m with randomness ρ is represented as $\mathsf{Enc}(m; \rho) = (1 + n)^m \rho^n \bmod n^2$.

It is easily verified that Paillier encryption scheme is additively homomorphic, such that $\mathsf{Enc}(m_1; \rho_1)\mathsf{Enc}(m_2; \rho_2) = \mathsf{Enc}(m_1 + m_2; \rho_1\rho_2)$ and $\mathsf{Enc}(m; \rho)^x = \mathsf{Enc}(xm; \rho^x)$.

2.3 Pedersen Commitment

Pedersen commitment scheme is used as a component of our protocol. Given a storng RSA modulus n, we can expect that there is a reasonably small value $k = \mathcal{O}(\log(n))$, such that $kn + 1$ is a prime, and thus find a group $\mathbb{G}$ of order n. Let the commitment key to be $\mathsf{ck} = (g_0, g_1, \ldots, g_\ell, h)$, where $g_0, \ldots, g_\ell, h$ are all generators of $\mathbb{G}$.

We denote the Pedersen commitment algorithms for single values as Com_i for $i = 0, \ldots, \ell$. Committing a value m with randomness r by Com_i is via computing

$\mathsf{Com}_i(m; r) = g_i^m h^r$. We further denote as Com the general Pedersen commitment for vectors, and committing a vector $\boldsymbol{m}$ with randomness r is to compute $\mathsf{Com}(\boldsymbol{m}; r) = (\prod_{i=1}^{\ell} g_i^{m_i})h^r$. Note that vectors with less than ℓ entries can be committed by setting the remaining entries to 0.

Pedersen commitment is computationally binding under the *discrete logarithm assumption*, such that a non-uniform probabilistic polynomial-time (PPT) adversary cannot find two openings of the same commitment except a negligible probability. The commitment scheme is perfectly hiding, because no matter what value/vector is committed, the commitment is uniformly distributed in $\mathbb{G}$.

Clearly, Pedersen commitment scheme is additively homomorphic, such that $\mathsf{Com}(\boldsymbol{m_1}; r_1)\mathsf{Com}(\boldsymbol{m_2}; r_2) = \mathsf{Enc}(\boldsymbol{m_1}+\boldsymbol{m_2}; r_1+r_2)$ and $\mathsf{Enc}(\boldsymbol{m}; r)^x = \mathsf{Enc}(x\boldsymbol{m}; xr)$.

2.4 The Generalized Schwartz–Zippel Lemma

In the later proof of security, we will use the following generalized Schwartz–Zippel lemma from time to time.

Lemma 1 (Generalized Schwartz–Zippel). *Let p be a non-zero multi-variate polynomial of total degree $d \geq 0$ over a ring $\mathbb{R}$. Let $\mathbb{S} \subseteq \mathbb{R}$ be a finite set with $|\mathbb{S}| \geq d$, such that $\forall a \neq b \in \mathbb{R}$, $a - b$ is not a zero divisor. Then the probability of $p(x_1, \ldots, x_\ell) = 0$ for randomly chosen $x_1, \ldots, x_\ell \leftarrow_\$ \mathbb{S}$ is at most $\frac{d}{|\mathbb{S}|}$.*

Given two multi-variate polynomials p_1 and p_2 and random $x_1, \ldots, x_\ell \leftarrow_\$ \mathbb{S}^\ell$, $p_1(x_1, \ldots, x_\ell) - p_2(x_1, \ldots, x_\ell) = 0$ always holds if $p_1 = p_2$, whereas holds with the probability at most $\frac{\max(d_1, d_2)}{|\mathbb{S}|}$ if $p_1 \neq p_2$.

3 Multi-exponentiation with Encrypted Bases Argument

In this section, we give the formal description of MEB protocol, propose the main body of our improved MEB protocol, and prove that our protocol is secure.

Description

- Common Reference String: Pedersen commitment key $\mathsf{ck} = (g_0, g_1, \ldots, g_\ell, h)$.
- Word: $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_\ell) \in (\{0,1\}^\kappa)^\ell$, $\ell + 1$ Paillier ciphertexts A and $\boldsymbol{a} = (a_1, \ldots, a_\ell)$. The public key of the Paillier encryption scheme is n, and we denote $\mu = ||n||$. Note that $2^\ell = \mathcal{O}(\mu^c)$ and $\kappa = \mathcal{O}(\mu^c)$ for a large enough constant c.
- Statement: There are some $(m_i, \rho_i)_{i=1,\ldots,\ell}$ and ρ such that $a_i = \mathsf{Enc}(m_i; \rho_i)$ for all $i = 1, \ldots, \ell$ and $A = \mathsf{Enc}(\prod_{i=1}^{\ell} m_i^{\lambda_i}; \rho)$.
- Witness: ρ, $(m_i, \rho_i)_{i=1,\ldots,\ell}$.

Procedure

1. P picks $(r_1, \ldots, r_\ell) \leftarrow_\$ (\mathbb{Z}_n)^\ell$ and $r_M \leftarrow_\$ \mathbb{Z}_n$, computes commitments $c_i \leftarrow \mathsf{Com}_i(m_i; r_i)$ for $i = 1, \ldots, m$ and $C \leftarrow \mathsf{Com}_0(\prod_{i=1}^{\ell} m_i^{\lambda_i}; r_M)$. Then P sends $(c_i)_{i=1,\ldots,\ell}$, C to P. V will continue to interact with P if all $c_i \in \mathbb{G}$, $C \in \mathbb{G}$. Otherwise, V outputs reject.
   Then P proves for each i her knowledge of (m_i, r_i, ρ_i) and the knowledge of $(\prod_{i=1}^{\ell} m_i^{\lambda_i}, r_M, \rho)$, such that $c_i = \mathsf{Com}_i(m_i; r_i)$, $a_i = \mathsf{Enc}(m_i; \rho_i)$, $C = \mathsf{Com}_0(\prod_{i=1}^{\ell} m_i^{\lambda_i}; r_M)$, and $A = \mathsf{Enc}(\prod_{i=1}^{\ell} m_i^{\lambda_i}; \rho)$ using the batch equality argument introduced in Section 4.1.
2. Let $(\lambda_{ij})_{j=1,\ldots,\kappa}$ be the bit decomposition of λ_i, *i.e.*, $\lambda_i = \lambda_{i\kappa} \cdots \lambda_{i1}$. Both parties locally compute general Pedersen commitments
   $$c_{\boldsymbol{a_j}} \leftarrow \mathsf{Com}((m_i^{\lambda_{ij}})_{i=1,\ldots,\ell}; \sum_{i=1}^{\ell} \lambda_{ij} r_i)$$
   for $j \in \{1, \ldots, \kappa\}$ from commitments $(c_i)_{i=1,\ldots,\ell}$, and set $c_{\boldsymbol{b_\kappa}} \leftarrow c_{\boldsymbol{a_\kappa}}$.
   P computes for $j \in \{1, \ldots, \kappa - 1\}$
   $$c_{\boldsymbol{b_j}} \leftarrow \mathsf{Com}((m_i^{\sum_{k=j}^{\kappa} 2^{k-j} \lambda_{ik}})_{i=1,\ldots,\ell}; r_{\boldsymbol{b_j}})$$
   and for $j \in \{2, \ldots, \kappa\}$
   $$c_{\boldsymbol{c_j}} \leftarrow \mathsf{Com}((m_i^{\sum_{k=j}^{\kappa} 2^{k-j+1} \lambda_{ik}})_{i=1,\ldots,\ell}; r_{\boldsymbol{c_j}}) .$$
   where all $r_{\boldsymbol{b_j}}$ and $r_{\boldsymbol{c_j}}$ are uniformly sampled from $\mathbb{Z}_n$.
   We denote the committed vectors inside $c_{\boldsymbol{a_j}}$ as $\boldsymbol{a_j} = (m_1^{\lambda_{1j}}, \ldots, m_\ell^{\lambda_{\ell j}})$, inside $c_{\boldsymbol{b_j}}$ as $\boldsymbol{b_j} = (m_1^{\sum_{k=j}^{\kappa} 2^{k-j} \lambda_{1k}}, \ldots, m_\ell^{\sum_{k=j}^{\kappa} 2^{k-j} \lambda_{\ell k}})$, and inside $c_{\boldsymbol{c_j}}$ as $\boldsymbol{c_j} = (m_1^{\sum_{k=j}^{\kappa} 2^{k-j+1} \lambda_{1k}}, \ldots, m_\ell^{\sum_{k=j}^{\kappa} 2^{k-j+1} \lambda_{\ell k}})$, respectively.
   P sends $(c_{\boldsymbol{b_j}})_{j=1,\ldots,\kappa-1}$ and $(c_{\boldsymbol{c_j}})_{j=2,\ldots,\kappa}$ to V.
3. If all $c_{\boldsymbol{b_j}} \in \mathbb{G}$ and $c_{\boldsymbol{c_j}} \in \mathbb{G}$, V sends random challenges $x, y \leftarrow_\$ (\mathbb{Z}_n^*)^2$ to P. Otherwise, V outputs reject.
4. Both parties locally compute $c_{\boldsymbol{a'_j}} \leftarrow c_{\boldsymbol{a_j}}^{x^j}$ for $j \in \{1, \ldots, \kappa - 1\}$, $c_{\boldsymbol{b'_j}} \leftarrow c_{\boldsymbol{b_j}}^{x^{\kappa+j-2}}$ for $j \in \{2, \ldots, \kappa\}$, $c_{\boldsymbol{d}} \leftarrow \prod_{j=1}^{\kappa-1} c_{\boldsymbol{b_j}}^{x^j} \prod_{j=2}^{\kappa} c_{\boldsymbol{c_j}}^{x^{\kappa+j-2}}$, and $c_{\boldsymbol{-1}} \leftarrow \mathsf{Com}(\boldsymbol{-1}; 0)$.
   Meanwhile, P computes the committed vectors and randomness of $c_{\boldsymbol{a'_j}}$ via $\boldsymbol{a'_j} \leftarrow x^j \boldsymbol{a_j}$ and $r_{\boldsymbol{a'_j}} \leftarrow x^j \sum_{i=1}^{\ell} \lambda_{ij} r_i$, of $c_{\boldsymbol{b'_j}}$ via $\boldsymbol{b'_j} \leftarrow x^{\kappa+j-2} \boldsymbol{b_j}$ and $r_{\boldsymbol{b'_j}} \leftarrow x^{\kappa+j-2} r_{\boldsymbol{b_j}}$, and of $c_{\boldsymbol{d}}$ via $\boldsymbol{d} \leftarrow \sum_{j=1}^{\kappa-1} x^j \boldsymbol{b_j} + \sum_{j=2}^{\kappa} x^{\kappa+j-2} \boldsymbol{c_j}$ and $r_{\boldsymbol{d}} \leftarrow \sum_{j=1}^{\kappa-1} x^j r_{\boldsymbol{b_j}} + \sum_{j=2}^{\kappa} x^{\kappa+j-2} r_{\boldsymbol{c_j}}$.
   Furthermore, let us define a bilinear operation $*$ for a given variable y as $\boldsymbol{a} * \boldsymbol{b} = \sum_i a_i b_i y^i$.
   Then P proves to V the knowledge of $(\boldsymbol{a'_j}, r_{\boldsymbol{a'_j}})_{j=1,\ldots,\kappa-1}$, $(\boldsymbol{b'_j}, r_{\boldsymbol{b'_j}})_{j=2,\ldots,\kappa}$, $(\boldsymbol{c_j}, r_{\boldsymbol{c_j}})_{j=2,\ldots,\kappa}$, $(\boldsymbol{b_j}, r_{\boldsymbol{b_j}})_{j=2,\ldots,\kappa}$, $\boldsymbol{d}, r_{\boldsymbol{d}}$ such that
   $$c_{\boldsymbol{a'_j}} = \mathsf{Com}(\boldsymbol{a'_j}; r_{\boldsymbol{a'_j}}) , \quad c_{\boldsymbol{b'_j}} = \mathsf{Com}(\boldsymbol{b'_j}; r_{\boldsymbol{b'_j}}) , \quad c_{\boldsymbol{c_j}} = \mathsf{Com}(\boldsymbol{c_j}; r_{\boldsymbol{c_j}}) ,$$

$$c_{\boldsymbol{b_j}} = \mathsf{Com}(\boldsymbol{b_j}; r_{\boldsymbol{b_j}})\,, \quad c_{\boldsymbol{d}} = \mathsf{Com}(\boldsymbol{d}; r_{\boldsymbol{d}})\,, \quad \sum_{j=1}^{\kappa-1} \boldsymbol{a'_j} * \boldsymbol{c_{j+1}} + \sum_{j=2}^{\kappa} \boldsymbol{b'_j} * \boldsymbol{b_j} - \boldsymbol{1} * \boldsymbol{d} = 0\,.$$

using the zero argument introduced in Section 4.2.

5. If the zero argument is rejected, V outputs reject. Otherwise, P proves to V the knowledge of $\boldsymbol{b_1}$, $r_{\boldsymbol{b_1}}$, $M = \prod_{i=1}^{\ell} m_i^{\lambda_i}$ and r_M, such that

$$c_{\boldsymbol{b_1}} = \mathsf{Com}(\boldsymbol{b_1}; r_{\boldsymbol{b_1}})\,, \quad C = \mathsf{Com}_0(M, r_M)\,, \quad \prod_{i=1}^{\ell} b_{1i} = M$$

using the committed single value product (CSVP) argument introduced in Section 4.3.

The round complexity of the protocol can be reduced by executing the batch equality argument and the CSVP argument in parallel in Step 1, and hence the protocol is a 5-round protocol.

Theorem 1. *The* MEB *protocol above is a public-coin SHVZK argument of knowledge.*

Proof. The completeness of the protocol follows from the completeness of the underlying batch equality argument. According to the homomorphic property of Pedersen commitment scheme, we can verify that

$$c_{\boldsymbol{a'_j}} = c_{\boldsymbol{a_j}}^{x^j} = \mathsf{Com}((x^j m_i^{\lambda_{ij}})_{i=1,\dots,\ell}; x^j(\sum_{i=1}^{\ell} \lambda_{ij} r_i)) = \mathsf{Com}(\boldsymbol{a'_j}; r_{\boldsymbol{a'_j}})\,,$$

$$c_{\boldsymbol{b'_j}} = c_{\boldsymbol{b_j}}^{x^{\kappa+j-2}} = \mathsf{Com}(x^{\kappa+j-2}\boldsymbol{b_j}; x^{\kappa+j-2} r_{\boldsymbol{b_j}}) = \mathsf{Com}(\boldsymbol{b'_j}; r_{\boldsymbol{b'_j}})\,,$$

and

$$\begin{aligned} c_{\boldsymbol{d}} &= \prod_{j=1}^{\kappa-1} c_{\boldsymbol{b_j}}^{x^j} \prod_{j=2}^{\kappa} c_{\boldsymbol{c_j}}^{x^{\kappa+j-2}} \\ &= \mathsf{Com}(\sum_{j=1}^{\kappa-1} x^j \boldsymbol{b_j} + \sum_{j=2}^{\kappa} x^{\kappa+j-2} \boldsymbol{c_j}; \sum_{j=1}^{\kappa-1} x^j r_{\boldsymbol{b_j}} + \sum_{j=2}^{\kappa} x^{\kappa+j-2} r_{\boldsymbol{c_j}}) \\ &= \mathsf{Com}(\boldsymbol{d}; r_{\boldsymbol{d}})\,. \end{aligned}$$

It is easy to verify that $\boldsymbol{b_j} = \boldsymbol{a_j}\boldsymbol{c_{j+1}}$ for $j \in \{1, \dots, \kappa-1\}$, and $\boldsymbol{c_j} = \boldsymbol{b_j}\boldsymbol{b_j}$ for $j \in \{2, \dots, \kappa\}$. Thus, we have

$$\begin{aligned} &\sum_{j=1}^{\kappa-1} \boldsymbol{a'_j}\boldsymbol{c_{j+1}} + \sum_{j=2}^{\kappa} \boldsymbol{b'_j}\boldsymbol{b_j} - \boldsymbol{d} \\ &= \sum_{j=1}^{\kappa-1} x^j \boldsymbol{a_j}\boldsymbol{c_{j+1}} + \sum_{j=2}^{\kappa} x^{\kappa+j-2}\boldsymbol{b_j}\boldsymbol{b_j} - \sum_{j=1}^{\kappa-1} x^j \boldsymbol{b_j} - \sum_{j=2}^{\kappa} x^{\kappa+j-2}\boldsymbol{c_j} \\ &= \sum_{j=1}^{\kappa-1} x^j(\boldsymbol{a_j}\boldsymbol{c_{j+1}} - \boldsymbol{b_j}) + \sum_{j=2}^{\kappa} x^{\kappa+j-2}(\boldsymbol{b_j}\boldsymbol{b_j} - \boldsymbol{c_j}) = 0 \end{aligned}$$

Furthermore, given the random y, if $\boldsymbol{a}\boldsymbol{b} = \boldsymbol{c}$, the equation $\boldsymbol{a} * \boldsymbol{b} = \boldsymbol{1} * \boldsymbol{c}$ holds. This shows that

$$\sum_{j=1}^{\kappa-1} \boldsymbol{a'_j} * \boldsymbol{c_{j+1}} + \sum_{j=2}^{\kappa} \boldsymbol{b'_j} * \boldsymbol{b_j} - \boldsymbol{1} * \boldsymbol{d} = 0 \,.$$

Finally, since $b_{1i} = m_i^{\lambda_i}$, the equation $\prod_{i=1}^{\ell} b_{1i} = M$ is always satisfied.

For SHVZK, the simulator $\mathcal{S}$ first picks $(r_1, \ldots, r_\ell) \leftarrow_\$ (\mathbb{Z}_n)^\ell$ and $r_M \leftarrow_\$ \mathbb{Z}_n$, computes commitments $c_i \leftarrow \mathsf{Com}_i(0; r_i)$ for $i = 1, \ldots, \ell$ and $C \leftarrow \mathsf{Com}(0; r_M)$. Since Pedersen commitment is prefect hiding, the commitments $(c_i)_{i=1,\ldots,\ell}$ and C have the same distribution as that of the real execution. Then $\mathcal{S}$ runs the SHVZK simulator for the batch equality argument.

Given the challenge x and y, the simulator $\mathcal{S}$ picks $r_{\boldsymbol{b_j}} \leftarrow_\$ \mathbb{Z}_n$ for $j = 1, \ldots, \kappa-1$, and $r_{\boldsymbol{c_j}} \leftarrow \mathbb{Z}_n$ for $j = 2, \ldots, \kappa$, computes commitments $c_{\boldsymbol{b_j}} = \mathsf{Com}(\boldsymbol{0}; r_{\boldsymbol{b_j}})$ and $c_{\boldsymbol{c_j}} = \mathsf{Com}(\boldsymbol{0}; r_{\boldsymbol{c_j}})$, and computes $c_{\boldsymbol{a_j}}$, $c_{\boldsymbol{a'_j}}$, $c_{\boldsymbol{b'_j}}$, $c_{\boldsymbol{d}}$, and $c_{\boldsymbol{-1}}$ as in the real execution. Due to the prefect hiding property of Pedersen commitment scheme, these commitments are perfectly indistinguishable from the real execution. The simulator $\mathcal{S}$ then runs the SHVZK simulators for both the zero argument and the committed single value product argument.

Because the distributions of commitments are perfectly indistinguishable from the real execution and the underlying arguments are SHVZK, the simulated transcripts generated by $\mathcal{S}$ are perfectly indistinguishable from those of real executions.

Here we show that the protocol is witness-extended emulation. The emulator runs the batch equality argument with witness-extended emulation to extract the encrypted values and randomness inside Paillier ciphertexts $(a_i)_{i=1,\ldots,\ell}$ and A. Since x and y are randomly choosen, according to Lemma 1, the equation $\sum_{j=1}^{\kappa-1} \boldsymbol{a'_j} * \boldsymbol{c_{j+1}} + \sum_{j=2}^{\kappa} \boldsymbol{b'_j} * \boldsymbol{b_j} - \boldsymbol{1} * \boldsymbol{d} = 0$ holds if $\boldsymbol{b_j} = \boldsymbol{a_j}\boldsymbol{c_{j+1}}$ and $\boldsymbol{c_j} = \boldsymbol{b_j}\boldsymbol{b_j}$, while holds with a negligible probability if there exists one equation that does not hold. Hence, based on the witness-extended emulation of the underlying batch equality argument, zero argument, and committed single value product argument, the extracted witness satisfies the statement of MEB with an overwhelming probability. □

4 Subprotocols

In this section, we present the subprotocols mentioned in Section 3.

4.1 Batch Equality Argument

Informally, the batch equality argument is to prove that a set of Paillier ciphertexts encrypt values inside a set of Pedersen commitments. We illustrate the batch equality argument in the following.

Description

– Common Reference String: Pedersen commitment key $\mathsf{ck} = (g_0, g_1, \dots, g_\ell, h)$.
– Word: The public key of the Paillier encryption scheme is n, and we denote $\mu = ||n||$. ℓ Pedersen commitments $\boldsymbol{c}$, and ℓ Paillier ciphertexts $\boldsymbol{a}$, where $2^\ell = \mathcal{O}(\mu^c)$ for a large enough constant c.
– Statement: There exist some $(m_i)_{i=1,\dots,\ell}$, $(r_i)_{i=1,\dots,\ell}$, and $(\rho_i)_{i=1,\dots,\ell}$, such that $c_i = \mathsf{Com}_i(m_i; r_i)$ and $a_i = \mathsf{Enc}(m_i; \rho_i)$ for $i = 1, \dots, \ell$.
– Witness: $(m_i)_{i=1,\dots,\ell}$, $(r_i)_{i=1,\dots,\ell}$, and $(\rho_i)_{i=1,\dots,\ell}$.

Procedure

1. P picks $\boldsymbol{u} \leftarrow_\$ (\mathbb{Z}_n)^\ell$, $\boldsymbol{v} \leftarrow_\$ (\mathbb{Z}_n)^\ell$, $\boldsymbol{w} \leftarrow_\$ (\mathbb{Z}_n^*)^\ell$, computes $x_i \leftarrow \mathsf{Com}_i(u_i, v_i)$ and $y_i \leftarrow \mathsf{Enc}(u_i; w_i)$ for $i = 1, \dots, \ell$, and sends $\boldsymbol{x}$, $\boldsymbol{y}$ to V.
2. If all $x_i \in \mathbb{G}$ and $y_i \in \mathbb{Z}_{n^2}^*$, V picks $(d, e) \leftarrow_\$ (\mathbb{Z}_n^*)^2$, and sends them to P. Otherwise, V outputs reject.
3. P computes $s \leftarrow \sum_{i=1}^\ell (v_i + r_i e) d^i \bmod n$, $t \leftarrow \prod_{i=1}^\ell (w_i \rho_i^e)^{d^i} \bmod n$, $z_i = (u_i + m_i e) d^i \bmod n$ for $i = 1, \dots, \ell$, and sends s, t, $\boldsymbol{z}$ to V.
4. V checks whether both $\mathsf{Com}(\boldsymbol{z}; s) = \prod_{i=1}^\ell (x_i c_i^e)^{d^i}$, $(\prod_{i=1}^\ell (1+n)^{z_i}) t^n \equiv \prod_{i=1}^\ell (y_i a_i^e)^{d^i} \bmod n^2$ hold and t is relatively prime to n. If all conditions hold, V outputs accept. Otherwise V outputs reject.

Theorem 2. *The batch equality argument protocol above is a public-coin SHVZK argument of knowledge.*

Proof. The completeness of the protocol can be verified as follows.

$$\begin{aligned}\mathsf{Com}(\boldsymbol{z}; s) &= \left(\prod_{i=1}^{\ell} g_i^{z_i}\right) h^s = \left(\prod_{i=1}^{\ell} g_i^{(u_i+m_ie)d^i}\right) h^{\sum_{j=1}^\ell (v_j + r_j e) d^j} \\ &= \prod_{i=1}^{\ell} (g_i^{u_i d^i} h^{v_i d^i} g_i^{m_i e d^i} h^{r_i e d^i}) = \prod_{i=1}^{\ell} (x_i c_i^e)^{d^i}\end{aligned}$$

$$\begin{aligned}(\prod_{i=1}^{\ell}(1+n)^{z_i}) t^n &\equiv (\prod_{i=1}^{\ell}(1+n)^{(u_i+m_ie)d^i})(\prod_{j=1}^{\ell}(w_j \rho_j^e)^{d^j})^n \\ &\equiv \prod_{i=1}^{\ell}((1+n)^{u_i d^i} w_i^{n d^i})((1+n)^{m_i e d^i} \rho_i^{n e d^i}) \\ &\equiv \prod_{i=1}^{\ell}(y_i a_i^e)^{d^i} \bmod n^2\end{aligned}$$

For SHVZK, given e and d, the simulator $\mathcal{S}$ picks $s_i \leftarrow_\$ \mathbb{Z}_n$, $t_i \leftarrow_\$ \mathbb{Z}_n^*$, $z_i' \leftarrow_\$ \mathbb{Z}_n$ for $i = 1, \dots, \ell$, and computes $s \leftarrow \sum_{i=1}^\ell s_i d^i \bmod n$, $t \leftarrow \prod_{i=1}^\ell t_i^{d^i} \bmod n$, and $z_i \leftarrow z_i' d^i \bmod n$. $\mathcal{S}$ then computes $x_i \leftarrow g_i^{z_i} h^{s_i} c_i^{-e}$, $y_i \leftarrow (1+n)^{z_i} t_i^n a_i^{-e} \bmod n^2$ for $i = 1, \dots, \ell$. It is easy to check that the simulated transcript $(\boldsymbol{x}, \boldsymbol{y}, e, d, s, t, \boldsymbol{z})$ is perfectly indistinguishable from the transcript of a real execution.

To prove that the protocol has witness-extended emulation, we let the emulator rewind the challenge phase to obtain ℓ pairs of accepted transcripts with the same $\boldsymbol{x}$, $\boldsymbol{y}$. Meanwhile, each pair has different random $(d_{(j)})_{j=1,\ldots,\ell}$, and both transcripts in each pair are respectively with different random e and e'. We denote these pairs of accepted transcripts with index $j = 1, \ldots, \ell$ as follows.

$$(\boldsymbol{x}, \boldsymbol{y}, e, d_{(j)}, s_{(j)}, t_{(j)}, \boldsymbol{z}_{(j)}) \qquad (\boldsymbol{x}, \boldsymbol{y}, e', d_{(j)}, s'_{(j)}, t'_{(j)}, \boldsymbol{z}'_{(j)})$$

Note that the witness-extended emulator will make on average 2ℓ arguments, and hence it runs in expected polynomial time. Given these 2ℓ accepted transcripts, we have for $j = 1, \ldots, \ell$ the equations

$$\mathsf{Com}(\boldsymbol{z}_{(j)}; s_{(j)}) = \prod_{i=1}^{\ell} (x_i c_i^e)^{d^i_{(j)}} , \qquad \mathsf{Com}(\boldsymbol{z}'_{(j)}; s'_{(j)}) = \prod_{i=1}^{\ell} (x_i c_i^{e'})^{d^i_{(j)}} .$$

The binding property of Pedersen commitment guarantees that with an overwhelming probability, there should be some $\boldsymbol{m}'$, $\boldsymbol{u}'$, such that for $j = 1, \ldots, \ell$, we have

$$\boldsymbol{z}_{(j)} = (u'_1 d_{(j)} + m'_m e d_{(j)}, \ldots, u'_m d^\ell_{(j)} + m'_m e d^\ell_{(j)})$$

and

$$\boldsymbol{z}'_{(j)} = (u'_1 d_{(j)} + m'_m e' d_{(j)}, \ldots, u'_m d^\ell_{(j)} + m'_m e' d^\ell_{(j)}) .$$

Given any such pairs, we can compute $\boldsymbol{m}'$, $\boldsymbol{u}'$ (*e.g.*, via Gaussian Elimination), since $(d^i_{(j)}, e d^i_{(j)})$ and $(d^i_{(j)}, e' d^i_{(j)})$ are linearly independent with an overwhelming probability for all $i = 1, \ldots, \ell$.

Moreover, the binding property of Pedersen commitment guarantees that there should be some $\boldsymbol{r}'$, $\boldsymbol{v}'$, such that for $j = 1, \ldots, \ell$,

$$s_{(j)} = \sum_{i=1}^{\ell} (v'_{(j)i} + r'_{(j)i} e) d^i_{(j)} \bmod n , \qquad s'_{(j)} = \sum_{i=1}^{\ell} (v'_{(j)i} + r'_{(j)i} e') d^i_{(j)} \bmod n .$$

Given ℓ such pairs, we can also recover $\boldsymbol{v}'$, $\boldsymbol{r}'$ with an overwhelming probability. Lemma 1 implies that these $\boldsymbol{v}'$, $\boldsymbol{r}'$ are the randomness of the commitments $\boldsymbol{x}$ and $\boldsymbol{c}$.

For these 2ℓ accepted transcripts, we also have for $j = 1, \ldots, \ell$ the equations

$$(\prod_{i=1}^{\ell} (1+n)^{z_{(j)i}}) t^n_{(j)} \equiv \prod_{i=1}^{\ell} (y_i a_i^e)^{d^i_{(j)}} \bmod n^2$$

and

$$(\prod_{i=1}^{\ell} (1+n)^{z_{(j)i}}) t^n_{(j)} \equiv \prod_{i=1}^{\ell} (y_i a_i^{e'})^{d^i_{(j)}} \bmod n^2 .$$

In the similar manner as above, the emulator can extract the same witness $\boldsymbol{m}'$, which is the encrypted values inside $(a_i)_{i=1,\ldots,\ell}$ with an overwhelming probability according to Lemma 1. Let $\alpha_i \leftarrow a_i (1+n)^{-m'_i} \bmod n^2$. There should be some $\boldsymbol{w}'$ and $\boldsymbol{\rho}'$, such that for $i = 1, \ldots, \ell$,

$$\alpha_i = \rho'^n_i \bmod n^2 ,$$

and for $j = 1, \ldots, \ell$,

$$t_{(j)}^n \equiv \prod_{i=1}^m (w_i' \rho_i'^e)^{d_{(j)}^i} \bmod n^2\,, \qquad t_{(j)}'^n \equiv \prod_{i=1}^m (w_i' \rho_i'^{e'})^{d_{(j)}^i} \bmod n^2\,.$$

The above first equation indexed by j divided by the second one is equal to

$$(t_{(j)} t_{(j)}'^{-1})^n \equiv \prod_{i=1}^m (\rho_i'^{e-e'})^{d_{(j)}^i} \bmod n^2\,.$$

Since $e - e'$ is relatively prime to n except a negligible probability, we can find β, γ, such that $\beta n + \gamma(e - e') = 1$. Let us denote $T_{(j)} = (\prod_{i=1}^m \alpha_i^{\beta d_{(j)}^i})(t_{(j)} t_{(j)}'^{-1})^{n\gamma}$, and thus we have

$$T_{(j)} = (\prod_{i=1}^m \alpha_i^{\beta d_{(j)}^i})(\prod_{i=1}^m (\rho_i'^{e-e'})^{d_{(j)}^i \gamma}) = \prod_{i=1}^m (\rho_i')^{d_{(j)}^i}\,.$$

We can use the Gaussian Elimination for exponents to compute $\boldsymbol{\rho}'$ from $T_{(j)} = \prod_{i=1}^m (\rho_i')^{d_{(j)}^i}$ for $j = 1, \ldots, \ell$. Since operations are over a group with hidden order, we here introduce how to reduce the exponent to 1, such that if $2^\ell = \mathcal{O}(\mu^c)$ for a large enough constant c, the emulator runs in polynomial time. For instance, if we want to reduce the exponent of ρ_1', *i.e.*, $d_{(j)}$, to 1, with a overwhelming probability we are able to find β' and γ', such that $\beta' n + \gamma' d_{(j)} = 1$. Then the emulator computes

$$\alpha^{\beta'} T_{(j)}^{\gamma'} = \rho_1' \sum_{i=2}^m (\rho_i')^{d_{(j)}^i \gamma'}$$

to achieve the goal.

According to Lemma 1, the extracted $\boldsymbol{\rho}'$ is the randomness of $(a_i)_{i=1,\ldots,\ell}$ except a negligible probability.

Hence, the protocol has witness-extended emulation. □

4.2 Zero Argument

For completeness, we restate the zero argument introduced in [1] as follows.

Description

- Common Reference String: Pedersen commitment key $\mathsf{ck} = (g_0, g_1, \ldots, g_\ell, h)$.
- Word: 2ℓ Pedersen general commitments $(c_{\boldsymbol{u_i}})_{i=1,\ldots,\ell}$, $(c_{\boldsymbol{v_i}})_{i=1,\ldots,\ell}$, a variable y, a bilinear map $*$.
- Statement: There exists some $(\boldsymbol{u_i}, r_{\boldsymbol{u_i}})_{i=1,\ldots,\ell}$, $(\boldsymbol{v_i}, r_{\boldsymbol{v_i}})_{i=1,\ldots,\ell}$, such that $c_{\boldsymbol{u_i}} = \mathsf{Com}(\boldsymbol{u_i}, r_{\boldsymbol{u_i}})$, $c_{\boldsymbol{v_i}} = \mathsf{Com}(\boldsymbol{v_i}, r_{\boldsymbol{v_i}})$ for all $i = 1, \ldots, \ell$, and $\sum_{i=1}^\ell \boldsymbol{u_i} * \boldsymbol{v_i} = 0$.
- Witness: $(\boldsymbol{u_i}, r_{\boldsymbol{u_i}})_{i=1,\ldots,\ell}$, $(\boldsymbol{v_i}, r_{\boldsymbol{v_i}})_{i=1,\ldots,\ell}$.

Procedure

1. P picks $\boldsymbol{u_0}, \boldsymbol{v_{\ell+1}} \leftarrow_\$ (\mathbb{Z}_n^\ell)^2$, $r_{\boldsymbol{u_0}}, r_{\boldsymbol{v_\ell}} \leftarrow_\$ (\mathbb{Z}_n)^2$, and computes
$$c_{\boldsymbol{u_0}} \leftarrow \mathsf{Com}(\boldsymbol{u_0}; r_{\boldsymbol{a_0}}), \qquad c_{\boldsymbol{v_{\ell+1}}} \leftarrow \mathsf{Com}(\boldsymbol{v_{\ell+1}}; r_{\boldsymbol{v_{\ell+1}}}).$$
Then P computes for $k = 0, \ldots, 2\ell$
$$d_k \leftarrow \sum_{\substack{0 \le i \le \ell, 1 \le j \le \ell+1 \\ j = \ell+1-k+i}} \boldsymbol{u_i} * \boldsymbol{v_j}.$$
P picks $(r_{d_0}, \ldots, r_{d_{2\ell}}) \leftarrow_\$ (\mathbb{Z}_n)^{2\ell+1}$, sets $r_{d_{\ell+1}} = 0$, and computes commitments $c_{d_k} = \mathsf{Com}_0(d_k; r_{d_k})$ for $k = 0, \ldots, 2\ell$. After the computation, P sends $c_{\boldsymbol{u_0}}$, $c_{\boldsymbol{v_{\ell+1}}}$, and $(c_{d_k})_{k=0,\ldots,2\ell}$ to V.
2. V sends $x \leftarrow_\$ \mathbb{Z}_n^*$ to P.
3. P computes
$$\boldsymbol{u} \leftarrow \sum_{i=0}^{\ell} x^i \boldsymbol{u_i} \quad r_{\boldsymbol{u}} \leftarrow \sum_{i=0}^{\ell} x^i r_{\boldsymbol{u_i}} \quad \boldsymbol{v} \leftarrow \sum_{j=1}^{\ell+1} x^{\ell-j+1} \boldsymbol{v_j} \quad r_{\boldsymbol{v}} \leftarrow \sum_{j=1}^{\ell+1} x^{\ell+1-j} r_{\boldsymbol{v_j}}$$
$$t \leftarrow \sum_{k=0}^{2\ell} x^k r_{d_k}$$
and sends $\boldsymbol{u}$, $r_{\boldsymbol{u}}$, $\boldsymbol{v}$, $r_{\boldsymbol{v}}$, t to V.
4. V outputs accept if $c_{\boldsymbol{u_0}} \in \mathbb{G}$, $c_{\boldsymbol{v_{\ell+1}}} \in \mathbb{G}$, $(c_{d_k})_{k=0,\ldots,2\ell} \in \mathbb{G}^{2\ell+1}$, $c_{d_{\ell+1}} = \mathsf{Com}_0(0;0)$, $\boldsymbol{u}, \boldsymbol{v} \in (\mathbb{Z}_n^\ell)^2$, $r_{\boldsymbol{u}}, r_{\boldsymbol{v}}, t \in (\mathbb{Z}_n)^3$, and
$$\prod_{i=0}^{\ell} c_{\boldsymbol{u_i}}^{x^i} = \mathsf{Com}(\boldsymbol{u}; r_{\boldsymbol{u}}), \quad \prod_{j=1}^{\ell+1} c_{\boldsymbol{v_j}}^{x^{\ell+1-j}} = \mathsf{Com}(\boldsymbol{v}; r_{\boldsymbol{v}}), \quad \prod_{k=0}^{2\ell} c_{d_k}^{x^k} = \mathsf{Com}_0(\boldsymbol{u} * \boldsymbol{v}; t).$$
Otherwise, V outputs reject.

Theorem 3 ([1]). *The zero argument protocol above is a public-coin SHVZK argument of knowledge.*

4.3 Committed Single Value Product (CSVP) Argument

We restate the committed single value product (CSVP) argument in [11] as follows.

Description

- Common Reference String: Pedersen commitment key $\mathsf{ck} = (g_0, g_1, \ldots, g_\ell, h)$.
- Word: A general Pedersen commitment c and a Pedersen commitment C committed by Com_0.
- Statement: There exits some $(\boldsymbol{m}, r)$ and (M, r_M), such that $c = \mathsf{Com}(\boldsymbol{m}; r)$, $C = \mathsf{Com}_0(M; r_M)$, and $M = \prod_{i=1}^{\ell} m_i$.
- Witness: $(\boldsymbol{m}, r)$ and (M, r_M).

Procedure

1. P computes
$$b_1 \leftarrow m_1\,, \quad b_2 \leftarrow m_1 m_2\,, \quad \cdots \quad b_\ell \leftarrow M\,.$$
Then P picks $d_1, \ldots, d_\ell, r_d, u \leftarrow_{\$} (\mathbb{Z}_n)^{\ell+2}$, sets $\delta_1 \leftarrow d_1$, $(\delta_2, \ldots, \delta_\ell) \leftarrow_{\$} (\mathbb{Z}_n)^{\ell-1}$, $r_\delta, r_\Delta \leftarrow_{\$} (\mathbb{Z}_n)^2$, computes
$$c_d \leftarrow \mathsf{Com}(\boldsymbol{d}; r_d)\,, \quad c_\delta \leftarrow \mathsf{Com}(-\delta_1 d_2, \ldots, -\delta_{\ell-1} d_\ell; r_\delta)\,, \quad a \leftarrow \mathsf{Com}_0(\delta_\ell; u)\,,$$
$$c_\Delta \leftarrow \mathsf{Com}(\delta_2 - m_2\delta_1 - b_1 d_2, \ldots, \delta_\ell - m_\ell \delta \ell - 1 - b_{\ell-1} d_\ell; r_\Delta)\,,$$
and sends c_d, c_δ, a, and c_Δ to V.
2. V sends the challenge $x \leftarrow_{\$} \mathbb{Z}_n^*$ to P.
3. P computes
$$m_1' \leftarrow x m_1 + d_1 \quad \cdots\,, \quad m_\ell' \leftarrow x m_\ell + d_\ell\,, \quad r' \leftarrow x r + r_d\,,$$
$$b_1' \leftarrow x b_1 + \delta_1 \quad \cdots\,, \quad b_\ell' \leftarrow x b_\ell + \delta_\ell\,, \quad s' \leftarrow x r_\Delta + r_\delta \quad z \leftarrow x r_M + u\,,$$
and sends $a_1', b_1', \ldots, a_\ell', b_\ell', r', s'$, z to V.
4. V outputs accept if $c_d, c_\delta, c_\Delta \in \mathbb{G}$, $a_1', b_1', \ldots, a_\ell', b_\ell', r', s', z \in \mathbb{Z}_n$ and
$$c^x c_d = \mathsf{Com}(\boldsymbol{m'}; r')\,, \quad c_\Delta^x c_\delta = \mathsf{Com}(x b_2' - b_1' m_2', \ldots, x b_\ell' - b_{\ell-1}' m_n'; s')\,,$$
$$C^x a = \mathsf{Com}_0(b_\ell'; z)\,, \quad b_1' = m_1'\,.$$
Otherwise, V outputs reject.

Theorem 4 ([11]). *The committed single value product (VSVP) argument protocol above is a public-coin SHVZK argument of knowledge.*

5 Evaluation and Comparisons

In this section, we compare our MEB protocol with the original MEB protocol introduced in [6]. from both theoretical and experimental aspects. We first analyze the argument size of both protocols and the number of communication rounds required by the protocols. Then, we conduct experiments to compare their running times in different settings of parameters.

5.1 Comparison on Argument Size and Communication Rounds

We denote the length of the bit-representation of the RSA modulus n as μ. Thus, elements in $\mathbb{Z}_n$ and $\mathbb{Z}_n^*$ can be represented by μ bits, and elements in $\mathbb{Z}_{n^2}$ can be represented by 2μ bits. We further denote the length of bit-representation of elements in $\mathbb{G}$ as η, and we can expect that $\eta = \mathcal{O}(\mu)$. Table 1 provides the comparison.

Table 1 presents the argument size and round complexity of all subprotocols of our protocol together with the overall complexities of both our protocol and the original MEB protocol. Both MEB protocols are of 5 rounds, while the size of ours is smaller than that of [6]. Since we can expect that $\eta \approx \mu$, the argument size of our protocol is roughly 36% smaller than that of protocol in [6]. Hence, our protocol has a lower communication cost compared with the original protocol.

Table 1. Comparison of argument size and communication rounds

Sub-protocols (Our MEB)	Argument size	Rounds
Batch equality argument	$(3\ell+4)\mu+(\ell+1)\eta$	3
Zero argument	$(2\ell+6)\mu+(4\kappa+1)\eta$	3
VSVP argument	$(2\ell+4)\mu+4\eta$	3
Main MEB argument	$2\mu+(\ell+2\kappa-2)\eta$	5
Overall Comparison:		
Our MEB protocol	$(7\ell+16)\mu+(2\ell+6\kappa+4)\eta$	5
MEB protocol in [6]	$(12\ell+19)\mu+(2\ell+6\kappa+15)\eta$	5

5.2 Experimental Results

We provide *proof-of-concept* implementations for both our protocol and the original protocol. The implementations are in `Python 3.8.1` on MacBook Air (2018) with 1.6 GHz dual-core Intel Core i5, 8GB of RAM, running macOS 10.15.3. We compare the running times of both protocols using different settings of parameters. Note that the communication cost is given in Section 5.1, and we here only measure the running times without the communication time. The results are shown in Table 2.

Table 2. Running time comparison of our MEB protocol and MEB protocol in [6]

μ	ℓ	t	Our MEB protocol			MEB protocol in [6]		
			Prover	Verifier	Total time	Prover	Verifier	Total time
1024	128	8	16.23s	5.16s	21.39s	18.78s	8.19s	26.97s
1024	128	32	44.28s	7.53s	51.81s	46.32s	9.64s	55.96s
1024	256	16	49.94s	12.72s	62.66s	53.19s	16.88s	70.07s
1024	256	32	88.33s	14.16s	102.49s	89.16s	17.66s	106.82s
1024	512	16	100.09s	26.48s	126.57s	102.47s	32.88s	135.30s
2048	128	8	104.32s	30.05s	134.37s	116.38s	56.58s	172.96s
2048	256	8	215.98s	71.86s	287.84s	223.99s	108.37s	332.36s

From Table 2, we can see that our protocol is more efficient for both prover and verifier compared with the original protocol. For all parameter settings, the computation done by the verifier in our protocol is roughly less than 80% of the computation done by the verifier in [6], and especially when $\mu = 2048$, $\ell = 128$ and $t = 8$, the execution time of the verifier in our protocol is 53% of that of the verifier in [6]. Therefore, our protocol saves more computation costs compared with the original protocol. We emphasize that the computation cost of the verifier is more critical for the example of computation on encrypted datasets, as we have mentioned in Section 1, since the data holder may serve multiple users and have more computational power, while users may use a device with much weaker computational capability. Hence, our improvement in the efficiency of the verifier is significant for this kind of applications.

6 Conclusions and Future Work

In this paper, we provide an improvement of the MEB protocol in both argument size and efficiency. We prove the security of our protocol and demonstrate our improvement from both theoretical and experimental aspects. Since MEB is the bottleneck for batching the executions of TCP and has advantages to be adopted in some applications as mentioned in Section 1, our improvement is significant for ESP, TCP-based protocols, and other applications.

Based on our results, future work could be carried out in two main directions. One direction is to further improve the MEB protocol in both communicated bits and efficiency. Since we only provide a *proof-of-concept* implementation with *single-thread* using a relatively slow programming language `Python`, the other direction is to have an optimized implementation of the protocol, which may further improve the performance of related cryptographic primitives and protocols. This may be more important for potential applications of the MEB protocol in practice.

References

1. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7237, pp. 263–280. Springer (2012)
2. Beaver, D.: Precomputing oblivious transfer. In: Coppersmith, D. (ed.) Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings. Lecture Notes in Computer Science, vol. 963, pp. 97–109. Springer (1995)
3. Beaver, D.: Adaptive zero knowledge and computational equivocation (extended abstract). In: Miller, G.L. (ed.) Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996. pp. 629–638. ACM (1996)
4. Castagnos, G., Imbert, L., Laguillaumie, F.: Encryption switching protocols revisited: Switching modulo p. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10401, pp. 255–287. Springer (2017)
5. Couteau, G., Peters, T., Pointcheval, D.: Secure distributed computation on private inputs. In: García-Alfaro, J., Kranakis, E., Bonfante, G. (eds.) Foundations and Practice of Security - 8th International Symposium, FPS 2015, Clermont-Ferrand, France, October 26-28, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9482, pp. 14–26. Springer (2015)

6. Couteau, G., Peters, T., Pointcheval, D.: Encryption switching protocols. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 308–338. Springer (2016)
7. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings. Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer (1986)
8. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009. pp. 169–178. ACM (2009)
9. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A.V. (ed.) Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA. pp. 218–229. ACM (1987)
10. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984)
11. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. J. Cryptology **23**(4), 546–579 (2010)
12. Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols - Techniques and Constructions. Information Security and Cryptography, Springer (2010)
13. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. J. Cryptology **16**(3), 143–184 (2003)
14. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding. Lecture Notes in Computer Science, vol. 1592, pp. 223–238. Springer (1999)
15. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings. Lecture Notes in Computer Science, vol. 576, pp. 129–140. Springer (1991)
16. Yao, A.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982. pp. 160–164. IEEE Computer Society (1982)
17. Yao, A.C.: How to generate and exchange secrets (extended abstract). In: 27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986. pp. 162–167. IEEE Computer Society (1986)