IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 30, 2019, accepted August 15, 2019, date of publication August 19, 2019, date of current version September 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2936210

Experimenting With Non-Interactive Range Proofs Based on the Strong RSA Assumption

MYUNGSUN KIM[1] AND HYUNG TAE LEE[2]

[1]Department of Information Security, College of ICT Convergence, University of Suwon, Hwaseong-si 18323, South Korea
[2]Division of Computer Science and Engineering, College of Engineering, Chonbuk National University, Jeonju 54896, South Korea

Corresponding author: Hyung Tae Lee (hyungtaelee@chonbuk.ac.kr)

The work of M. Kim was supported in part by the Institute for Information and communications Technology Promotion (IITP) Grant funded by the Korean Government (MSIT), Privacy-Preserving and Vulnerability Analysis for Smart Contract, under Grant 2018-0-00251. This work was done while H. T. Lee was with Nanyang Technological University, Singapore. The work of H. T. Lee was supported in part under Research Grant TL-9014101684-01, and in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) under Grant NRF-2018R1C1B6008476.

ABSTRACT Range proofs are proofs that a committed number m belongs to a range $[a, b]$ for public constants a, b, without leaking any information about the value m. In this work, we evaluate and analyze the performance of existing techniques for range proofs based on the strong RSA assumption while varying the range sizes. We first group the techniques into two classes. Our experiments show that the first class, being built on finding sums of squares (e.g., Groth's range proof), has sharply decreasing performance trends as the range size increases. Thus, solutions in this class seem to be useful primarily for small ranges. The second class, which relies on a direct proof (e.g., Boudot's range proof), exposes that the performance degradation slopes are not as steep as the range size grows, compared to solutions in the first group. However, this class's main drawback is that these methods require considerably more modular arithmetic than the first class. Concretely, the Groth and Boudot protocols achieve the best performance when the range sizes are less than and greater than 1410 bits, respectively. As part of this work, we consider an extension by combining the strong points of existing solutions and examine the result efficiency. Interestingly, however, our experimental results report that this extension outperforms either Groth's or Boudot's protocol for certain ranges, but there is no range for which the extension outperforms both.

INDEX TERMS Non-interactive zero-knowledge proof, range proof, strong RSA assumption.

I. INTRODUCTION

Zero-knowledge proofs have many applications as a key primitive in cryptographic constructions such as group signature [1], (security-enhanced) public-key encryption [2] and secure multiparty computation [3]. Particularly when employed in real-life applications, these examples demonstrate the usefulness of non-interactive zero-knowledge (NIZK) proofs, as they do not require connection-oriented interactions between participants.

In this work, we are interested in NIZK proofs for ranges that allow a prover to convince a verifier that a value m lies in a public set $\mathsf{R} = \{a, a+1, \cdots, b\}$ for $a \leq b \in \mathbb{Z}$ without revealing the value of m. Examples extensively relying on NIZK range proofs include smart contracts [4], anonymous credential [5], e-voting [6] and e-cash [7]. Since Brickell et al.'s first proposal [8], many solutions have been proposed to perform range proofs (e.g., [6], [9]–[16]). Among them, we restrict our interest to NIZK range proofs based on the strong RSA assumption [17], which is a widely adopted and well-studied cryptographic assumption.

There are two branches in this line of research: Boudot's approach [9] and Lipmma's approach [10]. Lipmma's range proof technique was later improved by Groth [6] and Couteau et al. [16].[1] To the best of our knowledge, there are no known works that comprehensively demonstrate which work is a better choice for a given range. The primary goal of this work is to answer the problem.

A. OVERVIEW OF OUR RESULTS

1) PROBING AND ASSESSING KNOWN SOLUTIONS

Recall that our motivation is to identify, among various range proof solutions, which solution performs best for a given

The associate editor coordinating the review of this article and approving it for publication was Yinghui Zhang.

[1]Precisely speaking, Couteau et al.'s range proof protocol [16] depends on the RSA assumption, not the strong RSA assumption. However, to simplify the comparison, throughout this paper, we do not consider the advantage of relying on a weaker assumption.

interval. To achieve this goal, we hone in on existing solutions through experimental study.

First, we review existing protocols for range proofs based on the strong RSA assumption and evaluate their performance characteristics from the prover perspective by varying the range size. Our starting point is the following question: ''How quickly can we find a sum of squares of a *large positive integer*, v?'' This is a natural question because the range proof techniques studied by Lipmaa [10] and their variants (e.g., [6] and most recently [16]) require executing an *online* algorithm to find such solutions. Thus, the algorithm's execution time is a key factor that heavily affects the overall performance, especially on the prover's side. Indeed, according to our experiments, the algorithm performance decreases considerably as the size of v increases. For example, in our experiments, while the algorithm finds the solutions within 10.3 ms for a v of 800 bits, it requires 200.0 ms for a v of 2000 bits.

Boudot's approach [9] can also be considered to prevent the heavy computation problem on the fly. However, a negative aspect of this protocol is that it requires a relatively large number of modular arithmetic on the prover's side. Allowing the commitment to an arbitrary integer, the sizes of the exponents in modular exponentiations may exceed the modulus size of range proof protocols. This is why our efficiency comparisons rely on the size of the exponents in modular exponentiations. From our analysis, when we assume that the output size of a deployed hash function is 2λ for a security parameter λ, the total exponent size of the prover in Boudot's protocol is $25|\mathsf{R}|+8\ell_n+140\lambda+32$ bits, where $|\mathsf{R}|$ is the target range size and ℓ_n is the bit size of the modulus of an exploited commitment scheme. However, the Groth protocol requires smaller exponent sizes than does the Boudot protocol. Concretely, the Groth protocol requires only $16|\mathsf{R}|+3\ell_n+54\lambda+13$ bits on the prover's side, and this quantity is always smaller than that of the Boudot protocol.

Accordingly, we may estimate that when the range size is relatively small, because the time to find a sum of squares has no significant impact on the prover's online run time, the Groth protocol outperforms all the other protocols on the prover's side. Reversely, we determine that, as the range size increases, an algorithm to find a sum of squares takes much longer to execute; in such cases, the Boudot protocol is faster than the Groth protocol. Indeed, our experimental results show that this performance reversal occurs at a size of approximately 1410 bits.

2) A HYBRID OF THE TWO

As a natural extension, we attempt to design an NIZK argument protocol for range proofs by integrating the features of the two branches of existing solutions based on the strong RSA assumption. (We call this extension Hybrid throughout the paper.) However, our experimental results unfortunately confirm that Hybrid outperforms either the Groth or Boudot protocol in certain ranges but does not outperform both simultaneously.

B. LITERATURE REVIEW

Since Brickell et al.'s solution [8], many range proofs have been proposed. As mentioned before, throughout this paper, we focus on strong RSA-based range proofs in the non-interactive manner. The key idea of protocols in this vein is the observation that if $m \in [a, b]$, then $(m-a)$ and $(b-m)$ should be non-negative, where $[a, b]$ denotes a closed interval between a and b, i.e., $[a, b] = \{m \in \mathbb{Z} : a \leq m \leq b\}$.

To prove that $m \in [a, b]$, Boudot's protocol [9] sets $v_1 = (m-a)$ and $v_2 = (b-m)$, and writes them as $v_1 = \alpha_1 + \alpha_2^2$ and $v_2 = \beta_1 + \beta_2^2$, where $\alpha_2 = \lfloor\sqrt{v_1}\rfloor$, $\alpha_1 = v_1 - \alpha_2^2$, and $\beta_2 = \lfloor\sqrt{v_2}\rfloor$, $\beta_1 = v_2 - \beta_2^2$. Here, $\lfloor x\rfloor$ denotes the largest integer that is less than or equal to x for a real number x. Then, the protocol uses two sub-protocols: One is to prove a square in commitments to α_2^2 and β_2^2, and the other is Chan, Frankel and Tsiounis's protocol (termed the CFT protocol for short) [18], which is used to prove that α_1, β_1 are bounded appropriately. Refer to Section III-A for the technical details of each sub-protocol.

The next consideration is Lipmaa's suggestion [10], in which the basic idea is to prove the non-negativity of integers as Boudot did. However, the Lipmaa protocol takes a different approach, using Lagrange's four-square theorem (a.k.a., Bachet's conjecture), which states that any positive integer can be written as a sum of four squares. Later, Groth [6] improved the Lipmaa protocol by exploiting Legendre's three-square theorem, which states that a positive integer v can be written as a sum of three squares if and only if v is not of the form $4^s(8t+7)$ for integers s and t. Most recently, Couteau et al. [16, Section 7.2] presented a variant of the Lipmaa protocol. Couteau et al.'s protocol proves the non-negativity of $v = (m-a)(b-m)$, instead of $m-a$ and $b-m$ each, by representing $4v+1$ as the sum of three squares.

Regarding the above protocols based on sums of squares, our main concern is that they should invoke an algorithm to find four (or three) squares whose sum is the given positive integer on the fly. In general, the Rabin-Shallit algorithm (RS algorithm for short) [19] is applied to find the four (or three) squares whereby their sum is a secret positive integer; however, this algorithm takes $O(|b-a|^4)$ time, where $|b-a|$ is the size of the range $[a, b]$. Lipmaa [10] introduced a somewhat improved algorithm in the RS algorithm to find the sum of four squares; however, the asymptotic complexities of both algorithms are the same. Consequently, solutions in this approach would suffer from performance bottlenecks in applications in which a large-sized interval should be used.

Indeed, this argument is confirmed by our implementation of the RS algorithm and Lipmaa's variant to find the sum of three and four squares, respectively (see Section IV-A). When the range size is 1200 bits, the average run times of the RS algorithm and Lipmaa's variant are approximately 33.9 ms and 33.5 ms, respectively. At the prover's side, the run times of the Groth protocol and the Lipmaa protocol are approximately 133.1 ms and 305.4 ms, respectively. These times mean that the algorithms account for 25.5% and 11.0% of the prover's total run times, respectively. However, when

the interval length increases to 1600 bits, the RS algorithm and Lipmaa's variant require approximately 91.0 ms and 87.0 ms, respectively, meaning that they consume approximately 35.3% and 21.6% of the total run times of the Groth and Lipmaa protocols, respectively. We note that because the Groth and Lipmaa protocols should run the RS algorithm and the Lipmaa's variant twice, respectively, in reality, the run times of the algorithms account for 70.6% and 43.2% of the total run times of the protocols, respectively, when the range size is 1600 bits.

1) A WAY OF COMBINING BOTH APPROACHES

For a secret integer m, if $m \in [a, b]$, then we have a non-negative integer $v = (m-a)(b-m)$. Integrating Boudot's and Couteau et al.'s ideas, one can express the integer v as $\alpha_1 + \alpha_2^2$ for some integers α_1 and α_2, and thus, we can prove the non-negativity of v by showing that α_1 is bounded and that α_2^2 is in a square. We expect that such an approach to prove non-negativity helps to reduce the number of modulus arithmetic operations.

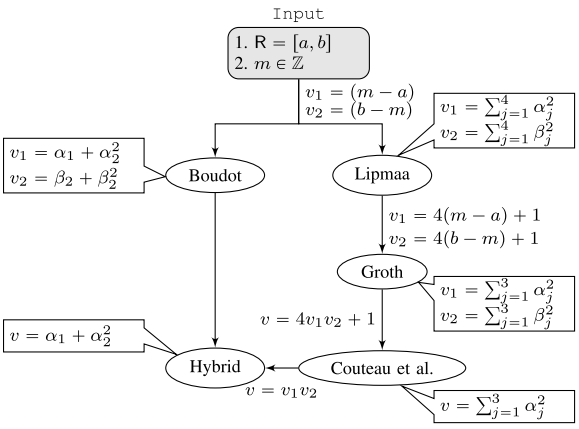


FIGURE 1. Key insights of the existing solutions.

In Fig. 1, we summarize the key ideas of the existing protocols and Hybrid with respect to the ways of decomposing non-negative integer(s) for a range proof.

C. OUTLINE OF THE PAPER

The remainder of this work is organized as follows. In Section II, we introduce the hardness assumption and building blocks common to known protocols. We then review representative strong RSA-based range proof protocols and explain a natural extension by combining two protocols among them in Section III. Our experimental results, along with detailed evaluations, are given in Section IV.

II. CRYPTOGRAPHIC ASSUMPTIONS AND PRIMITIVES

In this section, we briefly review the cryptographic assumptions and primitives used in known solutions.

1) NOTATION

Throughout this paper, for a set X, we use $x \xleftarrow{\$} X$ to denote that an element x is sampled uniformly at random from X. For $i, j \in \mathbb{Z}$, we denote the closed interval, $\{m \in \mathbb{Z} | i \leq m \leq j\}$, by $[i, j]$ and the open interval, $\{m \in \mathbb{Z} | i < m < j\}$, by (i, j). For a real number x, $\lfloor x \rfloor$ denotes the largest integer that is less than or equal to x.

A function $\nu : \mathbb{N} \to \mathbb{R}$ is negligible in λ if for all positive polynomials $f(\cdot)$ and sufficiently large λ, $\nu(\lambda) \leq \frac{1}{f(\lambda)}$. We use $\mathsf{poly}(\lambda)$ to represent unspecified polynomials in λ. A probabilistic polynomial-time (PPT) algorithm is a randomized algorithm that runs in $\mathsf{poly}(\lambda)$ time.

A. CRYPTOGRAPHIC ASSUMPTIONS

Let λ be a security parameter. Let G be an algorithm that takes a pair of λ and $\mathsf{b} \in \{0, 1\}$ as inputs and returns an RSA modulus n, a positive integer e such that $\gcd(e, \phi(n)) = 1$ for the Euler-totient function ϕ, and a random element $y \in \mathbb{Z}_n^*$ if $\mathsf{b} = 1$; otherwise, it returns an RSA modulus n and a random element $y \in \mathbb{Z}_n^*$. Throughout the paper, if implied by context, we sometimes omit explicitly writing "mod n" for modulo calculations of n.

1) THE FLEXIBLE RSA PROBLEM

The RSA problem can be stated as follows: Given a triple of values $(n, e, y) \leftarrow \mathsf{G}(1^\lambda, 1)$, find $x \in \mathbb{Z}_n^*$ such that $x^e = y \pmod n$. The flexible RSA problem is defined as follows.

Definition 1 (Flexible RSA Problem): Given a pair of values (n, y) obtained by invoking the algorithm $\mathsf{G}(1^\lambda, 0)$, find $e > 1$ and $x \in \mathbb{Z}_n^*$ such that $x^e = y \pmod n$.

2) THE STRONG RSA ASSUMPTION

The cryptographic primitives (e.g., commitment) in this section are based on the strong RSA assumption, which informally states that the flexible RSA problem is hard to solve. Since it was first introduced in [20], the strong RSA assumption has been used in constructing various cryptographic schemes (e.g., [17], [21]). This is considered as a stronger assumption than the RSA assumption (i.e., informally, the RSA problem is harder to solve than the strong RSA problem); however, currently, the only known way to break the assumptions is to solve the integer factorization problem.

Definition 2 (Strong RSA Assumption): The strong RSA assumption holds relative to G if for any PPT algorithm $\mathcal{A}$,

$$\Pr\left[(e > 1) \wedge (x^e = y) \middle| (n, y) \leftarrow \mathsf{G}(1^\lambda, 0); (x, e) \leftarrow \mathcal{A}(n, y)\right]$$

is negligible in the security parameter λ.

B. CRYPTOGRAPHIC TOOLS

1) INTEGER COMMITMENT

Existing range proofs under the strong RSA assumption use homomorphic integer commitment schemes. Only a few schemes are known in the literature (e.g., [6], [17], [22]), and all of them are very similar in structure.

Our implementations use the Fujisaki-Okamoto (FO) commitment scheme, which was first proposed in [17] and then revised by Damgård and Fujisaki [22]. Upon input of a

security parameter λ, we generate an RSA modulus n of bit length $\ell_n = \mathsf{poly}(\lambda)$ by generating two safe primes $p = 2p' + 1$ and $q = 2q' + 1$ for primes p', q' and setting $n = pq$. We select a random element g of order $p'q'$ (this step can be accomplished by selecting a random element in $\mathbb{Z}_n^*$ and squaring it). Then, we choose a random element x from $\{0,1\}^{\ell_r}$ for $\ell_r = \ell_n + \lambda$ and set $h = g^x$. Let $\mathbb{G}$ be a cyclic group generated by g. Throughout this paper, we use $\mathsf{desc}(\mathbb{G})$ to denote a description of the group $\mathbb{G}$–including the generator g and the element h. Finally, we set a commitment key $ck = (n, \mathsf{desc}(\mathbb{G}), \ell_n, \ell_r)$.

To commit a message m, we select an element $r \xleftarrow{\$} \{0,1\}^{\ell_r}$, and we compute $c = g^m h^r$. We write this as $c = \mathsf{CMT}_{g,h}(m, r)$. To verify the commitment c, if we reveal m, r, and μ, one checks if $c = \mu \cdot \mathsf{CMT}_{g,h}(m, r)$ and $\mu^2 = 1 \mod n$, and when both hold, it outputs 1; otherwise, it outputs 0. When we want to commit to multiple messages $m_1, \ldots, m_t$, we compute $c = \mathsf{CMT}_{g_1,\ldots,g_t,h}(m_1, \ldots, m_t; r) = g_1^{m_1} \cdots g_t^{m_t} h^r$ with a randomly chosen $r \xleftarrow{\$} \{0,1\}^{\ell_r}$, where all g_i are random generators of $\mathbb{G}$.

When additional bases (e.g., $\hat{g}, \hat{h}$, and g_i's) are needed for use, we implicitly assume that these values are known to the associated group description $\mathsf{desc}(\mathbb{G})$ before running protocols that have access to the commitment scheme.

2) NON-INTERACTIVE ZERO-KNOWLEDGE ARGUMENTS

A range proof is a typical example of the three-move honest verifier zero-knowledge argument. The prover has some statement y that she wants to prove, and she knows a witness x. She sends an announcement a, receives a random challenge e and responds with an answer z. Given (y, a, e, z), the verifier can now test whether to accept the argument.

Using the Fiat-Shamir transformation [23], we make the prover compute the challenge e as a hashed value of y and a. Specifically, the prover computes an argument (a, e, z), where $e = H(y, a)$ for a hash function $H : \{0,1\}^* \to \{0,1\}^{\ell_h}$ for sufficiently large integer ℓ_h. This is a de facto standard to make the argument non-interactive. Here, the hash function is modeled as a random function that pairs inputs (y, a) with a random output e. Furthermore, to argue zero-knowledge, we use a trick to program the random oracle. We refer the reader to [24, Chapter 4] for more details of theories about and practical examples of zero-knowledge proofs.

III. CASE STUDY: NIZK RANGE PROOF

We offer a compact specification of known results of the NIZK range proof based on the strong RSA assumption. As mentioned in the introduction, the literature includes four representative protocols: Boudot, Lipmaa, Groth, and Couteau et al. protocols. We additionally have some variants (e.g., [15] is a variant of Boudot's protocol); however, those variants are quite similar to the original protocols with respect to security and structure except that they attempted to adapt themselves to a specific setting. Thus, we will not consider such variants in our experiments. We begin with the Boudot protocol.

We assume that all NIZK protocols additionally take (implicitly) as input the system parameter ℓ_h, which indicates the output size of the exploited hash function.

Sub-protocol 1 PSE: Proof of the Same Exponent

Common input: $ck, c, \hat{c}$ and the randomness space $\mathcal{R}$

Prover's input: $m, r, \hat{r}$ such that $c = \mathsf{CMT}_{g,h}(m, r)$ and $\hat{c} = \mathsf{CMT}_{\hat{g},\hat{h}}(m, \hat{r})$

[**argument**]

1: choose $\nu \xleftarrow{\$} \mathcal{R}$, $r_1 \xleftarrow{\$} \{0,1\}^{\ell_r+\ell_h+\lambda}$, and $r_2 \xleftarrow{\$} \{0,1\}^{\ell_r+\ell_h+\lambda}$
2: compute $c_1 = \mathsf{CMT}_{g,h}(\nu, r_1)$ and $c_2 = \mathsf{CMT}_{\hat{g},\hat{h}}(\nu, r_2)$
3: compute $e = H(c_1 \parallel c_2)$ /* H: a hash function */
4: compute $\delta_0 = \nu + em$, $\delta_1 = r_1 + er$, $\delta_2 = r_2 + e\hat{r}$
5: output the argument $\pi = (e, \delta_0, \delta_1, \delta_2)$

[**verification**]

6: check if $e \stackrel{?}{=} H(g^{\delta_0} h^{\delta_1} c^{-e} \parallel \hat{g}^{\delta_0} \hat{h}^{\delta_2} \hat{c}^{-e})$

A. THE BOUDOT PROTOCOL

The Boudot protocol consists of three sub-protocols. The first sub-protocol is an NIZK protocol to prove that two commitments have the same message and the second sub-protocol is an NIZK protocol for proving that a committed value is a square of an integer. The last sub-protocol is an NIZK protocol for boundedness. For the last sub-protocol, the author utilized the protocol studied by Chan, Frankel and Tsiounis (CFT) in [18] without modifications. Throughout this paper, we denote the first sub-protocol by

$$\mathsf{PSE}\left(m, r, \hat{r} | c = \mathsf{CMT}_{g,h}(m, r) \wedge \hat{c} = \mathsf{CMT}_{\hat{g},\hat{h}}(m, \hat{r})\right),$$

the second sub-protocol by

$$\mathsf{PSQ}\left(m, r | c = \mathsf{CMT}_{g,h}(m^2, r)\right),$$

and the last sub-protocol by

$$\mathsf{CFT}(m, r | c = \mathsf{CMT}_{g,h}(m, r) \wedge m \in [L, U]).$$

For the first sub-protocol PSE, various methods have been proposed and have attempted to argue that two commitments retain the same secret value, e.g., see [8], [9], [18], [25]. We now describe a concrete example of NIZK argument protocols for the proof of the same exponent in [9], as exploited in our implementations in Section IV. Note that although we do not provide a concrete description of the randomness space $\mathcal{R}$ in the protocol, it should be at least $2^{\ell_h+\lambda}$-times larger than the public set to which the committed value m belongs for ensuring the security of the protocol, where λ is the security parameter and ℓ_h is the hash output size.

Next, a specific construction of PSQ can be attained by calling the protocol PSE as a sub-routine. Given a commit-

Sub-protocol 2 CFT: Proof of the Boundedness

Common input: $ck, c, [-\theta, \theta]$, and the randomness space $\mathcal{R}_0$

Prover's input: m, r such that $c = \mathsf{CMT}_{g,h}(m, r) \wedge m \in [-\theta, \theta]$

[**argument**]

1: choose $\nu \xleftarrow{\$} [0, \theta - 1]$ and $\gamma \xleftarrow{\$} \{0,1\}^{\ell_r+\ell_h+\lambda}$
2: compute $c_1 = \mathsf{CMT}_{g,h}(\nu, \gamma)$
3: compute $e = H(c_1)$
4: compute $\delta_0 = \nu + em$ and $\delta_1 = \gamma + er$. If $\delta_0 \notin [e \cdot \theta/\epsilon, \theta - 1]$, go to Step 1
5: output the argument $\pi = (e, \delta_0, \delta_1)$

[**verification**]

6: check if $e \stackrel{?}{=} H(g^{\delta_0} h^{\delta_1} c^{-e})$ and $\delta_0 \stackrel{?}{\in} [e \cdot \theta/\epsilon, \theta - 1]$

ment c to m, consider a commitment $\hat{c}$ to m by computing the power of c by m. Then, $\hat{c}$ can be thought of as a commitment to m^2 to base g. Thus, an NIZK argument to prove that a committed integer is a square can be written as

$$\mathsf{PSE}(m, r, \hat{r} | c = \mathsf{CMT}_{g,h}(m, r) \wedge \hat{c} = \mathsf{CMT}_{c,h}(m, \hat{r})) \quad (\ddagger)$$

where $\hat{c} = c^m h^{\hat{r}} = (\mathsf{CMT}_{g,h}(m, r))^m h^r = g^{m^2} h^{mr+\hat{r}}$. Clearly, $\hat{c}$ is a commitment to m^2. In conclusion, running PSE as in $(\ddagger)$ results in $\mathsf{PSQ}(m, r' | \hat{c} = \mathsf{CMT}_{g,h}(m^2, r'))$ with $r' = mr + \hat{r}$.

Finally, the Boudot protocol exploits the CFT protocol, which proves that, given a commitment c to m, the absolute value of m is less than θ, i.e., $m \in [-\theta, \theta]$, although m has to be in $[0, \theta/\epsilon]$ to achieve zero knowledge, where $\epsilon = 2^{\ell_h+\lambda}$ is an expansion rate.

The CFT protocol for this is presented in Sub-protocol 2. Because the CFT protocol has an expansion rate, the Boudot protocol cannot prove the exact boundness that it has to show using the CFT protocol without any modifications. To overcome this issue, when it would like to show $m \in [a, b]$, the Boudot protocol expands the target range from $[a, b]$ to $[A, B]$, where $A = 2^T a$ and $B = 2^T b$ for a proper integer T. Thereafter, if we show that $M = 2^T m$ is in $[2^T a - \theta', 2^T b + \theta']$ for some parameter $\theta' < 2^T$, this guarantees that $m \in (a - 1, b + 1)$ and that it is equivalent to $m \in [a, b]$ because m is an integer.

The same issue occurs in the hybrid protocol presented in Section III-C. Refer to [9] and Section III-C of this paper for the details on why the expanded range is required.

We provide the description of the Boudot protocol in Protocol 3. The Boudot protocol, using the CFT protocol at Step 8, guarantees that

$$-\theta' \le \tilde{M}_2, \hat{M}_2 \le \theta'.$$

Protocol 3 Boudot's NIZK Range Proof

Common input: a commitment c and the commitment key ck

Prover's input: m and r such that $c = \mathsf{CMT}_{g,h}(m, r)$ for $m \in \mathsf{R} = [a, b]$ and randomizer r

[**argument**]

1: set $T = 2(\ell_h + \lambda + 1) + |\mathsf{R}|$
2: set $\theta' = 2^{\ell_h+\lambda+T/2+1}\sqrt{b-a}$, $A = 2^T a$, and $B = 2^T b$
3: compute $C = c^{2^T} = \mathsf{CMT}_{g,h}(M, R)$, where $M = 2^T m$ and $R = 2^T r$
4: compute $\tilde{C} = Cg^{-A}$, $\hat{C} = g^B C^{-1}$
5: set $\tilde{M} = M - A$ and $\hat{M} = B - M$, and compute $\tilde{M}_1 = \lfloor\sqrt{\tilde{M}}\rfloor$, $\tilde{M}_2 = \tilde{M} - \tilde{M}_1^2$ and $\hat{M}_1 = \lfloor\sqrt{\hat{M}}\rfloor$, $\hat{M}_2 = \hat{M} - \hat{M}_1^2$
6: compute $\tilde{C}_1 = \mathsf{CMT}_{g,h}(\tilde{M}_1^2, \tilde{R}_1)$, $\tilde{C}_2 = \mathsf{CMT}_{g,h}(\tilde{M}_2, \tilde{R}_2)$ and $\hat{C}_1 = \mathsf{CMT}_{g,h}(\hat{M}_1^2, \hat{R}_1)$, $\hat{C}_2 = \mathsf{CMT}_{g,h}(\hat{M}_2, \hat{R}_2)$, where $\tilde{R}_1 \xleftarrow{\$} \{0,1\}^{\ell_r}$ and $\hat{R}_1 \xleftarrow{\$} \{0,1\}^{\ell_r}$, and $\tilde{R}_2, \hat{R}_2$ are set so that $\tilde{R}_1 + \tilde{R}_2 = R$, $\hat{R}_1 + \hat{R}_2 = -R$
7: compute proofs $\tilde{\pi}_1$ and $\hat{\pi}_1$ by running

$$\mathsf{PSQ}(\tilde{M}_1, \tilde{R}_1 | \tilde{C}_1 = \mathsf{CMT}_{g,h}(\tilde{M}_1^2, \tilde{R}_1))$$

and

$$\mathsf{PSQ}(\hat{M}_1, \hat{R}_1 | \hat{C}_1 = \mathsf{CMT}_{g,h}(\hat{M}_1^2, \hat{R}_1)),$$

respectively
8: compute proofs $\tilde{\pi}_2$ and $\hat{\pi}_2$ by running

$$\mathsf{CFT}(\tilde{M}_2, \tilde{R}_2 | \tilde{C}_2 = \mathsf{CMT}_{g,h}(\tilde{M}_2, \tilde{R}_2) \wedge \tilde{M}_2 \in [-\theta', \theta'])$$

and

$$\mathsf{CFT}(\hat{M}_2, \hat{R}_2 | \hat{C}_2 = \mathsf{CMT}_{g,h}(\hat{M}_2, \hat{R}_2) \wedge \hat{M}_2 \in [-\theta', \theta']),$$

respectively
9: output $\pi = (C, \tilde{C}, \hat{C}, \tilde{\pi}_1, \hat{\pi}_1, \tilde{\pi}_2, \hat{\pi}_2)$

[**verification**]

10: set $T = 2(\ell_h + \lambda + 1) + |\mathsf{R}|$
11: set $A = 2^T a$ and $B = 2^T b$, and compute $C = c^{2^T} \mod n$
12: compute $\tilde{C} = Cg^{-A}$, $\hat{C} = g^B C^{-1}$ and verify if they are equal to corresponding commitments in π
13: fetch $\tilde{C}_1, \hat{C}_1$ from $\tilde{\pi}_1, \hat{\pi}_1$, and verify $\tilde{\pi}_1$ and $\hat{\pi}_1$ to test the validity of the commitments to a square
14: compute $\tilde{C}_2 = \tilde{C}/\tilde{C}_1$ and $\hat{C}_2 = \hat{C}/\hat{C}_1$
15: verify $\tilde{\pi}_2$ and $\hat{\pi}_2$ to test if the committed integers $\tilde{M}_2, \hat{M}_2$ are in $[-\theta, \theta]$, letting $\theta' = 2^{\ell_h+\lambda+T/2+1}\sqrt{b-a}$

Because $M - A \ge \tilde{M}_2 \ge -\theta'$ and $B - M \ge \hat{M}_2 \ge -\theta'$, we have

$$A - \theta' \le M \le B + \theta'.$$

Specifically, if we set the parameters T and θ' appropriately so that $\theta' < 2^T$, it holds that

$$2^T(a-1) < M = 2^T m < 2^T(b+1),$$

and thus, $a \le m \le b$.

B. THE LIPMAA PROTOCOL

It is well known from number theory that $0 \le m \in \mathbb{Z}$ if and only if it can be written as $m = \sum_{i=1}^{4} \alpha_i^2$ with $\alpha_i \in \mathbb{Z}$. Lipmaa's protocol [10] uses this fact to prove that $m-a \ge 0$ and $b-m \ge 0$. We present the Lipmaa protocol in Protocol 4.

Later, Groth [6] improved Lipmaa's protocol in terms of efficiency by writing $m-a$ and $b-m$ as integers $v_1 = 4(m-a)+1$ and $v_2 = 4(b-m)+1$, respectively, and then by expressing v_1 and v_2 as sums of three squares (i.e., $v_1 = \alpha_1^2 + \alpha_2^2 + \alpha_3^2$ and $v_2 = \beta_1^2 + \beta_2^2 + \beta_3^2$ with $\alpha_i, \beta_i \in \mathbb{Z}$). Then, the author applied the same technique as Lipmaa to prove that both v_1 and v_2 are such integers.

Most recently, Couteau et al. [16] suggested a way to improve Lipmaa's protocol while representing $v = (m-a)(b-m)$ as a sum of four squares. However, note that the primary goal of their work is to construct a cryptographic protocol under the RSA assumption rather than the strong RSA assumption. We omit full descriptions of the last two protocols (i.e., Groth and Couteau et al.'s protocols) because proof techniques in both cases are quite similar to Lipmaa's protocol.

C. HYBRID: A COMBINATION OF BOUDOT AND COUTEAU et al.'s PROTOCOLS

It is quite natural to consider a combination of Boudot and Couteau et al.'s ideas so that the resulting solution is expected to inherit their positive properties. Roughly speaking, the combination proves the non-negativity of $v = (m-a)(b-m)$ as Couteau et al.'s idea but in a different manner. To prove the positivity of v, as in the Boudot protocol, this protocol represents v as $\alpha_1 + \alpha_2^2$ for $\alpha_2 = \lfloor \sqrt{v} \rfloor$ and $\alpha_1 = v - \alpha_2^2$ and proves the squareness of α_2^2. Then, the protocol shows that α_1 is non-negative by the CFT protocol. However, the CFT protocol does not allow one to directly show that $\alpha_1 \in [0, b_0]$ for some integer b_0. To address this problem, we show that $2^{2T}\alpha_1 \in [-\vartheta, 2^{2T}b_0 + \vartheta]$ for $\vartheta < 2^{2T}$ with a proper value T, as in Boudot's protocol. Then, it holds that $\alpha_1 \in (-1, b_0+1) = [0, b_0]$.

For notational convenience, we use $\mathcal{R} = [A, B]$ to denote an extended range of $\mathsf{R} = [a, b]$ by setting $A = 2^T a$ and $B = 2^T b$ (i.e., $\mathcal{R} = [2^T a, 2^T b]$) and use $|\mathsf{R}|$ to denote the bit size of the cardinality of the input range R (i.e., $|\mathsf{R}| = \lceil \log_2(b-a) \rceil$). We now give a full description of the protocol in Protocol 5.

1) THE SIZE OF T

Because the verifier confirms the non-negativity of α_1 by checking the value δ_{α_1}, it cannot confirm whether α_1 belongs

---

Protocol 4 Lipmaa's NIZK Range Proof

Common input: a commitment c and the commitment key ck

Prover's input: $m-a$ and r such that $c = \mathsf{CMT}_{g,h}(m-a, r)$ for $m-a \ge 0$ and randomizer r

[**argument**]

1: compute $c_a = cg^{-a}$ and $c_b = g^b c^{-1}$
2: set $\alpha = m-a$ and $\beta = b-m$
3: find $\{\alpha_i\}_{i=1}^4$ and $\{\beta_i\}_{i=1}^4$ such that $\alpha = \sum_{i=1}^4 \alpha_i^2$ and $\beta = \sum_{i=1}^4 \beta_i^2$ using the Lipmaa algorithm
4: compute $c_i = \mathsf{CMT}_{g,h}(\alpha_i, r_i)$ and $d_i = \mathsf{CMT}_{g,h}(\beta_i, s_i)$ by choosing $r_i, s_i \xleftarrow{\$} \{0,1\}^{\ell_r}$ and for $i = 1, 2, 3, 4$
5: pick

$$\bar{\alpha}_i \xleftarrow{\$} \{0,1\}^{\frac{|\mathsf{R}|}{2}+\ell_h+\lambda},\ \bar{r}_i \xleftarrow{\$} \{0,1\}^{\ell_r+\ell_h+\lambda},\ \bar{r} \xleftarrow{\$} \{0,1\}^{\frac{|\mathsf{R}|}{2}+\ell_h+\ell_r+\lambda}$$
$$\bar{\beta}_i \xleftarrow{\$} \{0,1\}^{\frac{|\mathsf{R}|}{2}+\ell_h+\lambda},\ \bar{s}_i \xleftarrow{\$} \{0,1\}^{\ell_r+\ell_h+\lambda},\ \bar{s} \xleftarrow{\$} \{0,1\}^{\frac{|\mathsf{R}|}{2}+\ell_h+\ell_r+\lambda}$$

for $1 \le i \le 4$
6: compute $\bar{c}_i = \mathsf{CMT}_{g,h}(\bar{\alpha}_i, \bar{r}_i)$, $\bar{d}_i = \mathsf{CMT}_{g,h}(\bar{\beta}_i, \bar{s}_i)$ for $1 \le i \le 4$
7: compute $\bar{c} = \mathsf{CMT}_{c_1,c_2,c_3,c_4,h}(\bar{\alpha}_1, \bar{\alpha}_2, \bar{\alpha}_3, \bar{\alpha}_4, \bar{r})$ and $\bar{d} = \mathsf{CMT}_{d_1,d_2,d_3,d_4,h}(\bar{\beta}_1, \bar{\beta}_2, \bar{\beta}_3, \bar{\beta}_4, \bar{s})$
8: compute $\Delta_1 = H(\bar{c}_1, \bar{c}_2, \bar{c}_3, \bar{c}_4, \bar{c})$ and $\Delta_2 = H(\bar{d}_1, \bar{d}_2, \bar{d}_3, \bar{d}_4, \bar{d})$
9: compute $e_1 = H(c_1, c_2, c_3, c_4, \bar{c}_1, \bar{c}_2, \bar{c}_3, \bar{c}_4, \bar{c})$ and $e_2 = H(d_1, d_2, d_3, d_4, \bar{d}_1, \bar{d}_2, \bar{d}_3, \bar{d}_4, \bar{d})$
10: compute $z_i = e_1\alpha_i + \bar{\alpha}_i$, $u_i = e_1 r_i + \bar{r}_i$, $w_i = e_2\beta_i + \bar{\beta}_i$, $u_i = e_2 s_i + \bar{s}_i$ for $1 \le i \le 4$
11: compute $u = e_1(r - \sum_{i=1}^4 \alpha_i r_i) + \bar{r}$, $v = e_2(s - \sum_{i=1}^4 \beta_i s_i) + \bar{s}$
12: output the argument $\pi = ((c_i)_{i=1}^4, (d_i)_{i=1}^4, (\bar{c}_i)_{i=1}^4, (\bar{d}_i)_{i=1}^4, \bar{c}, \bar{d}, \Delta_1, \Delta_2, e_1, e_2, (z_i)_{i=1}^4, (w_i)_{i=1}^4, (u_i)_{i=1}^4, (v_i)_{i=1}^4, u, v)$

[**verification**]

13: compute $c_a = cg^{-a}$ and $c_b = g^b c^{-1}$
14: check whether the following holds:

$$e_1 \stackrel{?}{=} H(c_1, c_2, c_3, c_4, \bar{c}_1, \bar{c}_2, \bar{c}_3, \bar{c}_4, \bar{c})$$
$$e_2 \stackrel{?}{=} H(d_1, d_2, d_3, d_4, \bar{d}_1, \bar{d}_2, \bar{d}_3, \bar{d}_4, \bar{d})$$
$$\Delta_1 \stackrel{?}{=} H\left((g^{z_i}h^{u_i}c_i^{-e_1})_{i=1}^4, \left(\prod_{i=1}^4 c_i^{z_i}\right) h^u c^{-e_1}\right)$$
$$\Delta_2 \stackrel{?}{=} H\left((g^{w_i}h^{v_i}d_i^{-e_2})_{i=1}^4, \left(\prod_{i=1}^4 d_i^{w_i}\right) h^v d^{-e_2}\right)$$

Protocol 5 Hybrid

Common input: $ck, c, \mathsf{R} = [a, b], T, A = 2^T a$, and $B = 2^T b$

Prover's input: m, r such that $c = \mathsf{CMT}_{g,h}(m, r) \wedge m \in \mathsf{R}$

[**argument**]

1: compute $C = c^{2^T} \mod n$, and set $M = 2^T m$ and $R = 2^T r$

2: compute $C_A = Cg^{-A}$ and $C_B = g^B C^{-1}$

3: compute $\hat{C} = \mathsf{CMT}_{C_A,h}(B - M, \hat{R})$ with $\hat{R} \xleftarrow{\$} \{0,1\}^{\ell_r}$

4: compute a proof π_1 by executing

$$\pi_1 \leftarrow \mathsf{PSE}(B - M, -R, \hat{R} | C_B = \mathsf{CMT}_{g,h}(B - M, -R) \wedge \hat{C} = \mathsf{CMT}_{C_A,h}(B - M, \hat{R}))$$

where $\pi_1 = (e_1, \delta_{11}, \delta_{12}, \delta_{13})$

5: find α_1, α_2 such that $v = \alpha_1 + \alpha_2^2$ by setting $\alpha_2 = \lfloor \sqrt{v} \rfloor$ and $\alpha_1 = v - \alpha_2^2$, where $v = (M - A)(B - M)$

6: compute $\bar{C} = \mathsf{CMT}_{g_1,\ldots,g_4,h}(v, \alpha_1, \alpha_2, \Delta; \bar{r})$, where

$$\bar{r} \xleftarrow{\$} \{0,1\}^{\ell_r},$$
$$r_v \xleftarrow{\$} \{0,1\}^{2T+2|\mathsf{R}|+\ell_h+\lambda},$$
$$r_{\alpha_1}, r_{\alpha_2} \xleftarrow{\$} \{0,1\}^{T+|\mathsf{R}|+\ell_h+\lambda}$$

and $\Delta = r_v - r_{\alpha_1} - 2\alpha_2 r_{\alpha_2}$

7: compute $\bar{C}_r = \mathsf{CMT}_{g_1,\ldots,g_4,h}(r_v, r_{\alpha_1}, r_{\alpha_2}, -r_{\alpha_2}^2; r_{\bar{r}})$ with $r_{\bar{r}} \xleftarrow{\$} \{0,1\}^{\ell_r+\ell_h+\lambda}$

8: $C_{r_v} = \mathsf{CMT}_{g,h}(r_v, R_{\hat{R}})$ with $R_{\hat{R}} \xleftarrow{\$} \{0,1\}^{2T+|\mathsf{R}|+\ell_r+\ell_h+\lambda}$

9: compute $e = H(C \parallel \hat{C} \parallel \bar{C} \parallel \bar{C}_r \parallel C_{r_v})$

10: compute

$$\delta_v = r_v + ev,\ \delta_{\alpha_1} = r_{\alpha_1} + e\alpha_1,\ \delta_{\alpha_2} = r_{\alpha_2} + e\alpha_2,$$
$$\delta_{\bar{r}} = r_{\bar{r}} + e\bar{r},\ \delta_{\hat{R}} = R_{\hat{R}} + e(R(B - M) + \hat{R});$$

if $\delta_{\alpha_1} \notin [0, 2^{\ell_h}(2^\lambda + 1)(B - A)]$, then restart the protocol

11: output the argument $\pi = (e, C, C_A, C_B, \hat{C}, \bar{C}, \bar{C}_r, C_{r_v}, \pi_1, \delta_v, \delta_{\alpha_1}, \delta_{\alpha_2}, \delta_{\bar{r}}, \delta_{\hat{R}})$

[**verification**]

12: compute $C = c^{2^T} \mod n$ and check the validity of C_A and C_B

13: verify π_1 by checking if $e_1 \stackrel{?}{=} H(g^{\delta_{11}} h^{\delta_{12}} C_B^{-e_1} \parallel C_A^{\delta_{11}} h^{\delta_{13}} \hat{C}^{-e_1})$

14: check if $\delta_{\alpha_1} \stackrel{?}{\in} [0, 2^{\ell_h}(2^\lambda + 1)(B - A)]$

15: check if $e \stackrel{?}{=} H(C \parallel \hat{C} \parallel \bar{C} \parallel \bar{C}_r \parallel C_{r_v})$

16: check whether $\hat{C}^e \cdot C_{r_v} \stackrel{?}{=} \mathsf{CMT}_{g,h}(\delta_v, \delta_{\hat{R}})$

17: check whether $\bar{C}^e \cdot \bar{C}_r \stackrel{?}{=} \mathsf{CMT}_{g_1,\ldots,g_4,h}(\delta_v, \delta_{\alpha_1}, \delta_{\alpha_2}, \delta_\Delta; \delta_{\bar{r}})$, where $\delta_\Delta = e(\delta_v - \delta_{\alpha_1}) - (\delta_{\alpha_2})^2$

to the exact set that it wants to check. More precisely, assume that the verifier wants to check whether $\alpha_1 \in [l, u]$ for the integers l and u. If $\alpha_1 \in [l, u]$, then we have

$$0 \le \delta_{\alpha_1} = r_{\alpha_1} + e\alpha_1 \le 2^{\ell_h} u + 2^{\ell_h+\lambda} u \quad (1)$$

because e is the output of a hash function and $r_{\alpha_1} \xleftarrow{\$} [0, 2^{\ell_h+\lambda} u]$. Therefore, the verifier can check whether $\delta_{\alpha_1} \stackrel{?}{\in} [0, 2^{\ell_h} u + 2^{\ell_h+\lambda} u]$. However, this does not guarantee that $\alpha_1 \in [l, u]$. If $0 \le \delta_{\alpha_1} \le 2^{\ell_h} u + 2^{\ell_h+\lambda} u$, then the verifier guarantees only that

$$-2^{\ell_h+\lambda} u \le \alpha_1 = (\delta_{\alpha_1} - r_{\alpha_1})/e \le 2^{\ell_h} u + 2^{\ell_h+\lambda} u. \quad (2)$$

Thus, we can prove only that $v \ge -2^{\ell_h+\lambda} u$ and not that $v \ge 0$. To address this problem, we enlarge the range of R by 2^T and consider the non-negativity of $v = 2^{2T}(m - a)(b - m)$ instead of $(m - a)(b - m)$. To this end, we set $A = 2^T a$, $B = 2^T b$, and $M = 2^T m$.

Let $v = (M - A)(B - M)$, which has a maximum value of $\left(\frac{B-A}{2}\right)^2$ at $M = \frac{A+B}{2}$. Because $\alpha_2 = \lfloor \sqrt{v} \rfloor$ and $\alpha_1 = v - \alpha_2^2$,

$$\alpha_2 = \lfloor \sqrt{v} \rfloor < \frac{B - A}{2}$$

and

$$\alpha_1 = v - \alpha_2^2 < v - (\sqrt{v} - 1)^2 = 2\sqrt{v} - 1 < B - A.$$

Thus, if we set $l = 0$ and $u = B - A$ in relation (1), we obtain $v = \alpha_1 + \alpha_2^2 \ge -2^{\ell_h+\lambda}(B - A)$ from relation (2). Because v is an integer, if $2^{\ell_h+\lambda}(B - A) < 2^{2T}$, then

$$v = 2^{2T}(m - a)(b - m) > -2^{2T},$$

which means that $v \ge 0$. Hence, T should satisfy

$$2^{\ell_h+\lambda}(B - A) \le 2^{\ell_h+\lambda} \cdot 2^{T+|\mathsf{R}|} < 2^{2T},$$

and consequently, $T > |\mathsf{R}| + \ell_h + \lambda$. Therefore, we set $T = |\mathsf{R}| + \ell_h + \lambda + 1$.

D. COMPREHENSIVE ANALYSIS

Now, we compare the computational and communication costs of Hybrid with Groth and Boudot's protocols, which achieve the best performance in the two branches.

1) COMPUTATIONAL COSTS

Existing range proof protocols consist mainly of modular exponentiations, hash evaluations, and integer multiplications. The prover and verifier require relatively large numbers of modular exponentiations, while only a small number of hash evaluations and integer multiplications is required. The exponent sizes of the modular exponentiations in range proof protocols can be larger than the modulus size. Thus, it does not make sense that the number of modular exponentiations of the protocol correctly indicates the computational cost of the protocol. To resolve this issue, we measured the costs in terms

TABLE 1. Comparison of computational costs (in bits).

Protocols	Prover		Verifier
	Size of Exponents	SoS	Size of Exponents
Boudot [9]	$25\|\mathsf{R}\| + 8\ell_n + 140\lambda + 32$	×	$15\|\mathsf{R}\| + 6\ell_n + 114\lambda + 32$
Groth [6]	$16\|\mathsf{R}\| + 3\ell_n + 54\lambda + 13$	○	$9\|\mathsf{R}\| + 2\ell_n + 51\lambda + 11$
Hybrid	$42\|\mathsf{R}\| + 6\ell_n + 111\lambda + 23$	×	$28\|\mathsf{R}\| + 4\ell_n + 93\lambda + 28$

SoS: use of algorithm for finding sum of squares, $|\mathsf{R}|$: target range size, ℓ_n: the modulus size for the FO commitment scheme, λ: security parameter.

of the total exponent bit size of the modular exponentiation required.

Recall that the RSA modulus size is ℓ_n bits, the randomizer size is ℓ_r bits, the output size of the hash function is ℓ_h bits, and the input range size is $|\mathsf{R}|$ bits. We set $\ell_r = \ell_n + \lambda$ and $\ell_h = 2\lambda$ for the security parameter λ.

Table 1 provides a comparison of the computational costs of representative protocols with respect to the bit size of the total exponents in the modular exponentiations. From the table, clearly, the prover in the Groth protocol requires the smallest exponents. However, the protocol needs to execute an algorithm to find the sum of squares; thus, it may require substantially more time if that algorithm's run time is large. In contrast, the computational efficiencies of the provers in Boudot's protocol and Hybrid rely on the input range size.

In general, if the input range $|\mathsf{R}|$ is small, then a prover using Hybrid is faster than one using the Boudot protocol. Otherwise, the prover in the Boudot protocol performs better. For example, when $\ell_n = 3072$ and $\lambda = 128$, the computational cost of a prover in Hybrid is less than that in Boudot's protocol when $|\mathsf{R}| < 580$ bits.

However, our implementation results are slightly different from the theoretical expectations. More specifically, the range that Hybrid outperforms Boudot's protocol is rather larger than the theoretical bound. This is because Hybrid's prover does 21 modular exponentiations, but Boudot's prover does 27 modular exponentiations, in total. When the total exponent is the same size, usually a small number of modular exponentiations with large exponents takes less time than a large number of modular exponentiations with small exponents. According to our experiments, when the modulus size is 3072 bits, 2 modular exponentiations with 2000-bit exponents require 2.877×10^7 CPU cycles, while 1 modular exponentiation with 4000-bit exponent requires 2.824×10^7 CPU cycles.

Meanwhile, the Groth protocol achieves the best performance on the verifier's side regardless of the size of the input range. In Section IV, we will discuss the details using a comprehensive implementation of the protocols examined above.

2) COMMUNICATION COSTS

Table 2 shows a comparison of the communication costs among the protocols studied so far. We confirm that the communication cost of the Groth protocol is the shortest. For example, when $\ell_n = 3072$ and $\lambda = 128$, the communication costs of the Groth, Boudot and Hybrid protocols are 4.2 KB, 7.8 KB and 7.8 KB, respectively, for a range size of 1536 bits (i.e., $|\mathsf{R}| = 1536$).

TABLE 2. Comparison of communication costs (in bits).

Protocols	Communication Cost
Boudot [9]	$8\|\mathsf{R}\| + 11\ell_n + 80\lambda + 22$
Groth [6]	$6\|\mathsf{R}\| + 5\ell_n + 37\lambda + 11$
Hybrid	$13\|\mathsf{R}\| + 11\ell_n + 53\lambda + 15$

$|\mathsf{R}|$: target range size, ℓ_n: the modulus size for the FO commitment scheme, λ: security parameter.

TABLE 3. Computational costs for finding a sum of squares (Test platform: Intel core i5 @ 3.4 GHz with 32 GB RAM).

Input Size	RS Algorithm		Lipmaa Algorithm	
	Time	Cycle†	Time	Cycle†
800 bits	10.3 ms	0.352	10.0 ms	0.341
1200 bits	33.9 ms	1.157	33.5 ms	1.144
1600 bits	91.0 ms	3.102	87.0 ms	2.966
2000 bits	200.0 ms	6.817	198.0 ms	6.747
2400 bits	360.5 ms	12.288	357.8 ms	12.195
2800 bits	623.5 ms	21.250	612.5 ms	20.875
3200 bits	925.7 ms	31.551	997.7 ms	34.005

†Cycle: $\times 10^8$

IV. EXPERIMENTS AND EVALUATION

In this section, we evaluate the effectiveness and efficiency of existing NIZK protocols for range proofs based on the strong RSA assumption. We first examine the CPU cycles and run times of algorithms for finding the sum of squares, which several protocols utilize as a key subroutine. We then provide detailed comparisons of existing representative protocols for range proofs in terms of CPU cycles and runtimes.

The testing platform was a modern PC with an Intel Core i5 CPU running at 3.4 GHz with 32 GB of DDR4 RAM. We employed NTL 11.3.2 [26] and GMP 6.1.2 [27] for efficient mathematical operations with large numbers.

A. EFFICIENCY OF FINDING A SUM OF SQUARES

We provide experimental results of the RS algorithm [19] and the Lipmaa algorithm [10] to find the sum of three and four squares. We remark that the RS algorithm is employed in the Groth and Couteau et al. protocols [6], [16], while the Lipmaa algorithm is utilized as a part of Lipmaa's range proof protocol [10].

Table 3 lists the run times and the CPU cycles of the two algorithms for various input sizes. All the results in the table are average values achieved from over 1000 randomly generated input values.

TABLE 4. Results of execution time comparisons based on CPU cycles and runtimes (Test platform: Intel core i5 @ 3.4 GHz with 32 GB RAM).

Protocols	800 bits				900 bits			
	Argument		Verification		Argument		Verification	
	Time	Cycle†	Time	Cycle†	Time	Cycle†	Time	Cycle†
Boudot [9]	149.1 ms	5.083	92.7 ms	3.159	153.7 ms	5.240	95.6 ms	3.258
Lipmaa [10]	243.3 ms	8.294	149.0 ms	5.078	255.4 ms	8.706	151.8 ms	5.174
Groth [6]	81.7 ms	2.785	42.6 ms	1.452	92.1 ms	3.140	42.8 ms	1.459
Couteau et al. [16]	173.7 ms	5.922	65.2 ms	2.223	226.7 ms	7.725	67.4 ms	2.298
Hybrid	138.0 ms	4.705	98.3 ms	3.349	146.6 ms	4.996	103.9 ms	3.543

Protocols	1000 bits				1100 bits			
	Argument		Verification		Argument		Verification	
	Time	Cycle†	Time	Cycle†	Time	Cycle†	Time	Cycle†
Boudot [9]	158.3 ms	5.396	98.6 ms	3.359	163.1 ms	5.558	101.6 ms	3.463
Lipmaa [10]	268.7 ms	9.159	154.7 ms	5.273	288.9 ms	9.846	157.7 ms	5.375
Groth [6]	102.7 ms	3.502	43.4 ms	1.482	117.9 ms	4.022	44.8 ms	1.527
Couteau et al. [16]	290.5 ms	9.902	69.6 ms	2.373	367.1 ms	12.512	71.9 ms	2.450
Hybrid	155.1 ms	5.287	109.7 ms	3.740	163.8 ms	5.582	115.5 ms	3.937

Protocols	1200 bits				1300 bits			
	Argument		Verification		Argument		Verification	
	Time	Cycle†	Time	Cycle†	Time	Cycle†	Time	Cycle†
Boudot [9]	167.5 ms	5.709	104.4 ms	3.558	172.0 ms	5.863	107.3 ms	3.658
Lipmaa [10]	305.4 ms	10.411	160.6 ms	5.475	331.2 ms	11.287	163.6 ms	5.577
Groth [6]	133.1 ms	4.539	46.3 ms	1.580	156.7 ms	5.340	48.0 ms	1.638
Couteau et al. [16]	461.2 ms	15.721	74.1 ms	2.526	566.1 ms	19.294	76.3 ms	2.600
Hybrid	172.1 ms	5.867	121.1 ms	4.129	180.6 ms	6.155	126.9 ms	4.326

Protocols	1400 bits				1500 bits			
	Argument		Verification		Argument		Verification	
	Time	Cycle†	Time	Cycle†	Time	Cycle†	Time	Cycle†
Boudot [9]	176.7 ms	6.021	110.2 ms	3.757	181.3 ms	6.179	113.2 ms	3.860
Lipmaa [10]	350.9 ms	11.962	166.6 ms	5.678	388.7 ms	13.250	169.8 ms	5.787
Groth [6]	175.2 ms	5.974	49.7 ms	1.695	212.9 ms	7.260	51.4 ms	1.753
Couteau et al. [16]	711.8 ms	24.261	78.6 ms	2.679	906.5 ms	30.895	80.7 ms	2.752
Hybrid	189.3 ms	6.451	132.9 ms	4.528	197.7 ms	6.738	138.4 ms	4.717

Protocols	1600 bits				1700 bits			
	Argument		Verification		Argument		Verification	
	Time	Cycle†	Time	Cycle†	Time	Cycle†	Time	Cycle†
Boudot [9]	186.2 ms	6.346	116.3 ms	3.946	190.5 ms	6.493	119.2 ms	4.062
Lipmaa [10]	421.2 ms	14.356	172.5 ms	5.878	457.0 ms	15.575	175.4 ms	5.978
Groth [6]	257.5 ms	8.778	53.0 ms	1.809	294.3 ms	10.034	54.7 ms	1.866
Couteau et al. [16]	1132.5 ms	38.597	82.9 ms	2.826	1230.6 ms	41.942	84.0 ms	2.863
Hybrid	206.2 ms	7.029	144.2 ms	4.915	214.7 ms	7.318	149.9 ms	5.110

†Cycle: $\times 10^8$

Table 3 shows the general trend that finding a solution for a larger input size requires more CPU time and cycles. More precisely, each algorithm takes 1.48–3.35 times longer as the input size increases by 400 bits. One may expect that the RS algorithm would be more efficient than the Lipmaa algorithm because the latter exploits the former as a subroutine. However, if an input value has the form $2^t \cdot (2k + 1)$ for some non-negative integers t and k, we see that the input size of the RS algorithm is reduced by approximately t bits. Consequently, the performance gap between the run times of the two algorithms depends on a specific form in which the input values are written as $2^t \cdot (2k + 1)$.

Our experiments show that the RS algorithm is slightly faster than the Lipmaa algorithm when the input size is 3200 bits, whereas the Lipmaa algorithm is slightly faster for other cases. However, the differences in the results for all input bit sizes are very narrow, and they can be reversed in each experiment.

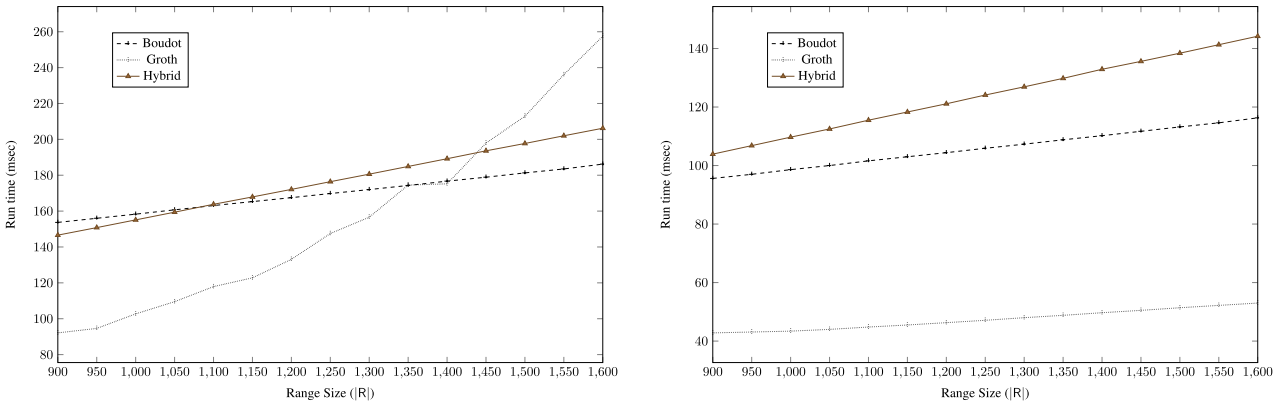
B. COMPARISONS AND EVALUATIONS

We continue by providing detailed comparisons of the prover's online run times and CPU cycles using the existing protocols while varying the range sizes. We set the

TABLE 5. Comparison of the selected protocols by narrowing down a measuring interval size (Test platform: Intel core i5 @ 3.4 GHz with 32 GB RAM).

Input Size	Argument						Verification					
	Boudot		Groth		Hybrid		Boudot		Groth		Hybrid	
	Time	Cycle†	Time	Cycle†	Time	Cycle†	Time	Cycle†	Time	Cycle†	Time	Cycle†
900 bits	153.7 ms	5.240	92.1 ms	3.140	146.6 ms	4.996	95.6 ms	3.258	42.8 ms	1.459	103.9 ms	3.543
950 bits	156.0 ms	5.318	94.6 ms	3.226	150.8 ms	5.141	97.0 ms	3.308	43.1 ms	1.469	106.8 ms	3.641
1000 bits	158.3 ms	5.396	102.7 ms	3.502	155.1 ms	5.287	98.6 ms	3.359	43.4 ms	1.482	109.7 ms	3.740
1050 bits	160.7 ms	5.477	109.5 ms	3.734	159.4 ms	5.434	100.0 ms	3.409	44.0 ms	1.500	112.5 ms	3.835
1100 bits	163.1 ms	5.558	117.9 ms	4.022	163.8 ms	5.582	101.6 ms	3.463	44.8 ms	1.527	115.5 ms	3.937
1150 bits	165.3 ms	5.633	122.8 ms	4.186	167.9 ms	5.723	103.0 ms	3.509	45.5 ms	1.551	118.3 ms	4.033
1200 bits	167.5 ms	5.709	133.1 ms	4.539	172.1 ms	5.867	104.4 ms	3.558	46.3 ms	1.580	121.1 ms	4.129
1250 bits	169.8 ms	5.789	147.5 ms	5.029	176.4 ms	6.014	105.9 ms	3.609	47.1 ms	1.608	124.1 ms	4.229
1300 bits	172.0 ms	5.863	156.6 ms	5.340	180.6 ms	6.155	107.3 ms	3.658	48.0 ms	1.638	126.9 ms	4.326
1350 bits	174.3 ms	5.942	174.5 ms	5.951	184.9 ms	6.302	108.8 ms	3.708	48.8 ms	1.666	129.8 ms	4.424
1400 bits	176.7 ms	6.021	175.2 ms	5.974	189.2 ms	6.451	110.2 ms	3.757	49.7 ms	1.695	132.9 ms	4.528
1450 bits	178.9 ms	6.097	197.9 ms	6.747	193.6 ms	6.598	111.7 ms	3.807	50.5 ms	1.724	135.6 ms	4.621
1500 bits	181.3 ms	6.179	212.9 ms	7.260	197.7 ms	6.738	113.2 ms	3.860	51.4 ms	1.753	138.4 ms	4.717
1550 bits	183.5 ms	6.253	236.1 ms	8.051	202.0 ms	6.883	114.6 ms	3.907	52.2 ms	1.780	141.3 ms	4.817
1600 bits	186.2 ms	6.346	257.5 ms	8.778	206.2 ms	7.029	116.3 ms	3.964	53.0 ms	1.809	144.2 ms	4.915

†Cycle: $\times 10^8$



(a) Run times on the prover's side

(b) Run time on the verifier's side

FIGURE 2. Run time comparisons of the selected protocols.

size of an RSA modulus n to 3072 bits to achieve 128-bit security [28].

Table 4 shows experimental results for the existing protocols for range proofs with respect to various range sizes. The table rows show the run times and CPU cycles of Boudot's protocol [9], Lipmaa's protocol [10], Groth's protocol [6],[2] Couteau et al.'s protocol [16], and Hybrid described in Section III-C. All the results in the table are average values calculated over 1000 randomly generated input values by executing 10 parameter generations of the FO commitment scheme and generating 100 input values for each parameter.

[2]In [6], Groth presented an NIZK protocol for positivity by modifying Lipmaa et al.'s protocol [29] with an algorithm that finds the sum of three squares. Groth's protocol can easily be applied for range proofs; however, he did not give detailed specifications for such a protocol. To perform a range proof of a message m in an interval $\mathsf{R} = [a, b]$, we obtain the protocol by setting $v_0 = (m - a)$, $v_1 = (b - m)$, and $M = b$ as shown in Fig. 4 of [6].

Together with Table 3 in Section IV-A, Table 4 tells us that Lipmaa's, Groth's, and Couteau et al.'s protocols require considerably more time to generate arguments than does Boudot's protocol or Hybrid as the range size increases. This result occurs because of the run times required to find the sums of three or four squares. Note that Lipmaa's and Groth's protocols invoke the Lipmaa algorithm and the RS algorithm, respectively, twice for each experiment. Compared with the results in Table 3, we observe that finding the sum of squares for one target element requires 20.7%, 35.4%, and 81.7% of the prover's run time in Lipmaa, Groth, and Couteau et al.'s protocols, respectively, when the range size is 1600 bits. Recall that in the Couteau et al. protocol, the input size of the RS algorithm is twice as long as the range size because it considers the non-negativity of $v = (m - a)(b - m)$ for $m \in [a, b]$. Furthermore, we see that the ratio between the time to find the sum of squares and the prover's total run time tends to increase rapidly as the range size increases. Thus,

their protocols do not seem suitable for large ranges (i.e., larger than 1600 bits).

Table 4 also shows that the Groth protocol achieves the best performance on the prover's side if the range size is less than 1400 bits, while the Boudot protocol achieves the best performance when the range size is greater than 1400 bits.

To take a closer look at the efficiency with respect to range size, we provide the experimental results of run times and CPU cycles under Boudot's and Groth's protocols and Hybrid by restricting to a specific range from 900 bits to 1600 bits; see Table 5. All the results in the table are average values calculated over 1000 randomly generated input values, as in Table 4. The results in Table 5 are graphically shown in Fig. 2a and Fig. 2b.

In summary, on the prover's side, if the range size is less than 1410 bits, the Groth protocol achieves the best performance; otherwise, the Boudot protocol achieves the best performance. We can observe that in the case that the range sizes are less than 1050 bits and larger than 1450 bits, the Hybrid protocol outperforms the Boudot and Groth protocols, respectively. Unfortunately, there is no range size such that the Hybrid protocol achieves the best performance. On the other hand, on the verifier's side, the Groth protocol runs about 2 times faster than the Boudot protocol and Hybrid.

As a side note, we remark that prover's run time of the Groth protocol in Fig. 2a increases quartic in the range size due to the quartic complexity of the RS algorithm. (See Section I-B for the complexity of the RS algorithm.)

V. OUR RESULT SUMMARY

In this work, we evaluated and analyzed the performance of existing protocols for range proofs based on the strong RSA assumption. For this purpose, we first classified existing representative protocols of this line into two branches. One branch includes the Lipmaa [10], Groth [6], and Couteau et al. protocols [16], which employ an algorithm to find the sums of squares as a sub-routine. The other branch includes Boudot's protocol [9], which does not employ this technique. We observed that the run time and CPU cycles for finding sums of squares in existing protocols consume substantial percentages of the prover's online processing time and CPU cycles, respectively, as the range size is increased. Thus, the former branch is suitable for small range sizes, e.g., less than 1410 bits at 128-bit security; however, the latter is better at larger range sizes. We also attempted to consider a natural extension by combining the advantages of the two branches. Unfortunately, the hybrid method outperforms either the former or latter branch for certain ranges, but not both branches simultaneously.

REFERENCES

[1] D. Chaum and E. van Heyst, "Group signatures," in *Proc. Advances in Cryptology—EUROCRYPT*. Brighton, U.K.: Springer, 1991, pp. 257–265.

[2] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attacks," in *Proc. ACM Symp. Theory Comput.*, May 1990, pp. 427–437.

[3] O. Goldreich, S. Micali, and A. Wigderson, "How to play ANY mental game," in *Proc. ACM Symp. Theory Comput.*, 1987, pp. 218–229.

[4] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE Symp. Secur. Privacy*, May 2016, pp. 839–858.

[5] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, Apr. 2001, pp. 93–118.

[6] J. Groth, "Non-interactive zero-knowledge arguments for voting," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2005, pp. 467–482.

[7] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, "Compact E-cash," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2005, pp. 302–321.

[8] E. F. Brickell, D. Chaum, I. Damgård, and J. van de Graaf, "Gradual and verifiable release of a secret," in *Proc. Conf. Theory Appl. Cryptograph. Techn.*, Dec. 2000, pp. 156–166.

[9] F. Boudot, "Efficient proofs that a committed number lies in an interval," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, May 2000, pp. 431–444.

[10] H. Lipmaa, "On diophantine complexity and statistical zero-knowledge arguments," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2003, pp. 398–415.

[11] A. Kiayias, Y. Tsiounis, and M. Yung, "Traceable signatures," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2004, pp. 571–589.

[12] S. Canard, I. Coisel, and J. Traoré, "Complex zero-knowledge proofs of knowledge are easy to use," in *Proc. Int. Conf. Provable Secur.*, 2007, pp. 122–137.

[13] J. Camenisch, R. Chaabouni, and A. Shelat, "Efficient protocols for set membership and range proofs," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2008, pp. 234–252.

[14] J. Groth, "Efficient zero-knowledge arguments from two-tiered homomorphic commitments," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2011, pp. 431–448.

[15] K. Peng and L. Yi, "Studying a range proof technique—Exception and optimisation," in *Proc. Int. Conf. Cryptol. Afr.*, 2013, pp. 328–341.

[16] G. Couteau, T. Peters, and D. Pointcheval, "Removing the strong RSA assumption from arguments over the integers," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2017, pp. 321–350.

[17] E. Fujisaki and T. Okamoto, "Statistical zero knowledge protocols to prove modular polynomial relations," in *Proc. Annu. Int. Cryptol. Conf.*, 1997, pp. 16–30.

[18] A. Chan, Y. Frankel, and Y. Tsiounis, "Easy come—Easy go divisible cash," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 1998, pp. 561–575.

[19] M. Rabin and J. Shallit, "Randomized algorithms in number theory," *Commun. Pure Appl. Math.*, vol. 39, no. S1, pp. S240–S256, 1986.

[20] N. Bari and B. Pfitzmann, "Collision-free accumulators and fail-stop signature schemes without trees," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 1997, pp. 480–494.

[21] R. Cramer and V. Shoup, "Signature schemes based on the strong RSA assumption," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 3, pp. 161–185, 2000.

[22] I. Damgård and E. Fujisaki, "A statistically-hiding integer commitment scheme based on groups with hidden order," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2002, pp. 125–142.

[23] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proc. Conf. Theory Appl. Cryptograph. Techn.*, 1986, pp. 186–194.

[24] O. Goldreich, *Buy Foundations of Cryptography*. Cambridge, U.K.: Cambridge Univ. Press, 2001.

[25] E. Fujisaki and T. Okamoto, "A practical and provably secure scheme for publicly verifiable secret sharing and its applications," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 1998, pp. 32–46.

[26] V. Shoup. *NTL: A Library for Doing Number Theory Version 11.3.2*. Accessed: Jul. 1, 2019. [Online]. Available: http://www.shoup.net/ntl/

[27] *GMP: The GNU Multiple Precision Arithmetic Library Version 6.1.2*. Accessed: Jul. 1, 2019. [Online]. Available: http://gmplib.org

[28] National Institute of Standards and Technology, *Recommendation for Key Management*, document 800-57, Jan. 2016.

[29] H. Lipmaa, N. Asokan, and V. Niemi, "Secure Vickrey auctions without threshold trust," in *Proc. Financial Cryptography*. Southampton, Bermuda: Springer, 2002, pp. 87–101.

MYUNGSUN KIM received the B.S. degree in computer science and engineering from Sogang University, Seoul, South Korea, in 1994, the M.S. degree in computer science and engineering from the Information and Communications University (ICU), Daejeon, in 2002, and the Ph.D. degree in mathematics from Seoul National University (SNU), Seoul, in 2012.

He is currently an Assistant Professor with the Department of Information Security, University of Suwon. His research interest includes multiparty computation in cryptography.

HYUNG TAE LEE received the B.S., M.S., and Ph.D. degrees in mathematics from Seoul National University, South Korea, in 2006, 2008, and 2013, respectively.

He was a research fellow with Nanyang Technological University, Singapore. He is currently an Assistant Professor with the Division of Computer Science and Engineering, College of Engineering, Chonbuk National University, South Korea. His research interests include computational number theory and cryptography.