Optimized Paillier's Cryptosystem with Fast Encryption and Decryption

Huanyu Ma*

Ant Financial Services Group
China

Shuai Han*

Ant Financial Services Group
China

Hao Lei†

Ant Financial Services Group
China

ABSTRACT

In this paper, we propose a new optimization for the Paillier's additively homomorphic encryption scheme (Eurocrypt'99). At the heart of our optimization is a well-chosen subgroup of the underlying $\mathbb{Z}_N^*$, which is used as the randomness space for masking messages during encryption. The size of the subgroup is significantly smaller than that of $\mathbb{Z}_N^*$, leading to faster encryption and decryption algorithms of our optimization. We establish the one-wayness and semantic security of our optimized Paillier scheme upon those of an optimization (i.e., "Scheme 3") made by Paillier in Eurocrypt'99. Thus, our optimized scheme is one-way under the partial discrete logarithm (PDL) assumption, and is semantically secure under the decisional PDL (DPDL) assumption. On the other hand, we present a detailed analysis on the concrete security of our optimized scheme under several known methods. To provide 112-bit security, our analysis suggests that a 2048-bit modulus N and a well-chosen subgroup of size 448-bit would suffice. We compare our optimization with existing optimized Paillier schemes, including the Jurik's optimization proposed by Jurik in his Ph.D. thesis and the Paillier's optimization in Eurocrypt'99. Our experiments show that, – the encryption of our optimization is about 2.7 times *faster* than that of the Jurik's optimization and is about 7.5 times *faster* than that of the Paillier's optimization;

– the decryption of our optimization is about 4.1 times *faster* than that of the Jurik's optimization and has a similar performance with that of the Paillier's optimization.

KEYWORDS

public-key encryption, homomorphic encryption, optimization, provable security

ACM Reference Format:

Huanyu Ma, Shuai Han, and Hao Lei. 2021. Optimized Paillier's Cryptosystem with Fast Encryption and Decryption. In *Annual Computer Security Applications Conference (ACSAC '21)*, December 6–10, 2021, Virtual Event, USA. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3485832.3485842>

*Both authors contributed equally to the paper

†Corresponding author

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike International 4.0 License.

ACSAC '21, December 6–10, 2021, Virtual Event, USA
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8579-4/21/12.
<https://doi.org/10.1145/3485832.3485842>

1 INTRODUCTION

Homomorphic encryption (HE) schemes enable computations on encrypted data. The operations on ciphertexts can be translated into operations on the underlying plaintexts. This property is extremely useful and has found numerous applications, such as e-voting [4, 24], threshold cryptosystems [17, 25], as well as modern applications including multi-party computation (MPC) [5, 19], outsourcing computation [23, 26] and blockchain [10, 13].

Generally, there are two classes of HE schemes. The first class is *partially* HE, which allows performing one type of homomorphic operation on encrypted data, usually either addition or multiplication. For example, the ElGamal scheme [22] is multiplicatively homomorphic while the Paillier scheme [33] is additively homomorphic. The second class includes *somewhat* HE (SWHE) and *fully* HE (FHE), which enable both addition and multiplication operations on encrypted data. It has been a long-lasting open problem to construct FHE that is homomorphic for arbitrary operations. In 2009, Gentry [28] proposed the first FHE scheme from ideal lattices. Since then, a lot of brilliant works [8, 9, 14, 21] (to name a few) have devoted to improving the efficiency of FHE or base FHE on more kinds of assumptions. However, the full-fledged FHE is still far from practical for large scale real applications.

Additively homomorphic encryption. Nevertheless, partially HE schemes, especially additively HE, are already quite useful in many practical settings. For example, in the setting of blockchain-based confidential payment mechanism [10], an additively HE scheme is employed to preserve the privacy of accounts while at the same time allows changing encrypted accounts without decrypting the ciphertexts first.

Another example is (semi-honest) multi-party computation, where an additively HE scheme could be applied to compute several useful statistical functions on encrypted data, such as counting, summation, (weighted) average, arithmetic mean and linear regression [3, 23]. In the case when both the data $X = \{x_i\}$ and its squares $X^2 = \{x_i^2\}$ are encrypted, one could compute more statistical functions including variance, standard deviation and root mean square (RMS). Furthermore, if another data $Y = \{y_i\}$ is provided in the clear, one could compute several useful statistical functions between X and Y , such as covariance, Pearson's and uncentered correlation coefficient, and the least squares fit of $\{(x_i, y_i)\}$ [3, 7, 23].

The de facto security requirement for an additively HE scheme is semantic security [29] (a.k.a. indistinguishability under chosen-plaintext attacks, or IND-CPA for short). It stipulates that, for any polynomial-time adversary that is given only the public key, it is infeasible to distinguish an encryption of message chosen by the adversary itself from an encryption of a uniformly random message.

The commonly-used additively HE schemes in practice include the “exponential” ElGamal scheme [16] and the Paillier scheme [33]. Recall that the original ElGamal scheme [22] is multiplicatively homomorphic. For the “exponential” ElGamal scheme [16], it encodes messages in the exponent, and requires solving a discrete logarithm to recover the message from the exponent in the decryption process. This limits the size of the message space. Typically, the “exponential” ElGamal scheme could only be used to encrypt messages of at most 32-bit [10]. This narrows the applications of the “exponential” ElGamal scheme in large settings.

In contrast, the Paillier scheme [33] admits a very large message space. It operates over an integral ring modulo a big integer N^2 , where $N = PQ$ with primes P, Q , and supports encrypting messages of at most 2048-bit if we instantiate N with a 2048-bit integer. The Paillier scheme is semantically secure under the decisional composite residuosity (DCR) assumption [33]. However, both the encryption and decryption algorithms of the Paillier scheme involve an expensive *non-fixed-base* modular exponentiation with a roughly 2048-bit exponent. More precisely, the encryption algorithm is as follows

$$\text{Enc}(pk = N, m \in \mathbb{Z}_N; r \in \mathbb{Z}_N^*) = (1 + m \cdot N) \cdot \boxed{\boxed{r^N}} \pmod{N^2},$$

and the decryption algorithm is

$$\text{Dec}(sk = \lambda, c \in \mathbb{Z}_{N^2}^*) = \frac{(\boxed{\boxed{c^\lambda \pmod{N^2}}} - 1)/N}{\lambda} \pmod{N}, \quad (1)$$

where $\lambda = \text{lcm}(P-1, Q-1)$ is of roughly 2048-bit, and “lcm” stands for least common multiple. We highlight the modular exponentiations with dotted boxes.

Optimizations for the Paillier scheme. To pursue fast encryption and/or fast decryption, several variants of the Paillier scheme were proposed in the literature.

In the same paper [33, Section 6], Paillier himself presented a variant with *fast decryption* and named it as “Scheme 3” (while the basic Paillier scheme described above was called “Scheme 1”). We use “PaiOpt” to denote the optimization made by Paillier. Concretely, a group element g of order αN is chosen from $\mathbb{Z}_{N^2}^*$ in the key generation process, with α an integer significantly smaller than λ . For example, α is suggested to be a 160-bit prime in [33] and a 320-bit product of two primes in [30]. The encryption algorithm of the Paillier’s optimization PaiOpt is as follows

$$\begin{aligned} \text{Enc}_{\text{PaiOpt}}(pk = (N, g), m \in \mathbb{Z}_N; r \in \mathbb{Z}_\alpha) \\ = g^{m+Nr} = \boxed{\boxed{g^m}} \cdot \boxed{\boxed{(g^N)^r}} \pmod{N^2}, \end{aligned}$$

and the decryption algorithm is

$$\text{Dec}_{\text{PaiOpt}}(sk = \alpha, c \in \mathbb{Z}_{N^2}^*) = \frac{(\boxed{\boxed{c^\alpha \pmod{N^2}}} - 1)/N}{(g^\alpha \pmod{N^2} - 1)/N} \pmod{N}.$$

For a 320-bit α , the decryption involves a *non-fixed-base* modular exponentiation with exponent α of only 320-bit. Thus, the decryption algorithm is roughly 6 times faster than that of the basic Paillier scheme [33]. The encryption algorithm involves two *fixed-base* modular exponentiations, one with exponent m and the other with a roughly 320-bit exponent r . To fully take advantage of the large message space, one might want to use the Paillier scheme (and its variants) to encrypt long messages. In the worst case, m might be

2048-bit, then the encryption involves *fixed-base* modular exponentiations with roughly (2048+320)-bit exponents. The Paillier’s optimization PaiOpt is semantically secure under the decisional partial discrete logarithm (DPDL) assumption [33]. We note that DPDL is a stronger assumption than DCR.

In 2003, Jurik [31, Section 2.3] described another variant of the Paillier scheme with *fast encryption*. (This variant was also included in an extended work [18, Section 4.1].) We use “JurOpt” to denote the optimization made by Jurik. Concretely, for $N = PQ$ with properly generated primes P and Q , the set $\mathbb{Z}_N^*[+1]$ containing all elements with Jacobi symbol $+1$ in $\mathbb{Z}_N^*$ constitutes a *cyclic* group of order λ , and a generator f of $\mathbb{Z}_N^*[+1]$ is chosen in the key generation process. The encryption algorithm of the Jurik’s optimization JurOpt is as follows

$$\begin{aligned} \text{Enc}_{\text{JurOpt}}(pk = (N, f), m \in \mathbb{Z}_N; r \in \mathbb{Z}_{2^{1024}}) \\ = (1 + m \cdot N) \cdot \boxed{\boxed{(f^N)^r}} \pmod{N^2}. \end{aligned}$$

Now the encryption involves a *fixed-base* modular exponentiation with exponent r of only 1024-bit. We note that one could speed-up the encryption by pre-computing powers of the fixed base (f^N). The decryption algorithm is identical to that of the basic Paillier scheme (i.e., (1)). The Jurik’s optimization JurOpt is semantically secure under the DCR assumption [31].

Catalano et al. [12] and Sakurai and Takagi [35] also proposed variants of the Paillier scheme. Unfortunately, their schemes lose the important property of additive homomorphism, thus are no longer suitable in the numerous applications described above. Shafagh et al. [36] suggested packing several small messages (such as 32-bit integers) into a long message (say 2048-bit) and then encrypt it using Paillier scheme. This might improve the amortized encryption performance. However, their scheme encrypts vectors of small messages, thus it is only entry-wisely additively homomorphic and only supports a limited number of homomorphic operations. See, e.g. [11], for a survey of existing variants of the Paillier scheme.

It seems to be a challenging task to design a variant of the Paillier scheme that significantly optimizes the encryption and decryption processes simultaneously, while retains the useful additively homomorphic property.

1.1 Our Contributions

In this work, we propose a new variant of the Paillier’s additively homomorphic encryption scheme, called “NewOpt”, which has both fast encryption and fast decryption algorithms. More precisely,

- We investigate the structure of certain subgroups of $\mathbb{Z}_N^*$, based on which we present a new optimization NewOpt for the Paillier scheme [33]. Our optimization employs a well-chosen subgroup of $\mathbb{Z}_N^*$ as the randomness space for masking messages during encryption. Our NewOpt inherits the additive homomorphism from the Paillier scheme.
- We establish the one-wayness and semantic security of our NewOpt upon that of PaiOpt (i.e., the Scheme 3 optimized by Paillier in [33]). Since PaiOpt is semantically secure under the DPDL assumption, so is our NewOpt. We note that DPDL is a stronger assumption than DCR.

Table 1: Comparison among the (basic) Paillier scheme and its additively homomorphic variants. Here n denote the bit-length of modulus N and $|m|$ the bit-length of plaintext m . In general, $|m|$ could be as large as n , and on average, $|m|$ is around $n/2$. Let ℓ denote the bit-length of α for the Paillier's optimization, and denote the bit-length of the size of $\mathbb{QR}_N^\beta$ for our optimization. Typically, ℓ is much smaller than n . “non-FBME” is short for non-fixed-base modular exponentiation and “FBME” fixed-base modular exponentiation. For “FBME”, the performance could be further improved by pre-computing powers of the fixed base (see Sect. 6 for more details).

		The basic Paillier (Scheme 1 in [33])	Jurik's optimization JurOpt [31]	Paillier's optimization PaiOpt (Scheme 3 in [33])	Our optimization NewOpt
algebraic structure	ciphertext form	$(1 + N)^m \cdot r^N$	$(1 + N)^m \cdot (f^N)^r$	$g^m \cdot (g^N)^r$	$(1 + N)^m \cdot (h^N)^r$
	randomness space	$r \in \mathbb{Z}_N^*$	$f \in \mathbb{Z}_N^*[+1]$	g generates a subgroup of $\mathbb{Z}_N^*$ of order αN	$h \in \mathbb{QR}_N^\beta \otimes \langle -1 \rangle$
security	provable security	semantically secure			
	assumption	DCR	DCR	DPDL	DPDL
	concrete security (112-bit security)	$n = 2048$	$n = 2048$	$n = 2048$ $\ell \geq 448$	$n = 2048$ $\ell \geq 448$
efficiency	encryption	non-FBME with n -bit exponent	FBME with $n/2$ -bit exponent	FBME with $(m + \ell)$ -bit exponent	FBME with ℓ -bit exponent
	decryption	non-FBME with n -bit exponent	non-FBME with n -bit exponent	non-FBME with ℓ -bit exponent	non-FBME with ℓ -bit exponent

- We present a detailed analysis on the concrete security of our NewOpt under several known methods, including certain integer factorization algorithms [1, 34], a specific integer factorization algorithm on small secret CRT-exponents [6], the baby-step giant-step algorithm [37] and the Coppersmith's method [15]. To provide 112-bit security, our analysis suggests that a 2048-bit modulus N and a well-chosen subgroup of size 448-bit are sufficient for our optimized scheme.
 - We give a comprehensive performance evaluation of our NewOpt, and compare it with the Jurik's optimization JurOpt [31] and the Paillier's optimization PaiOpt [33]. Our experiments show that
 - the key generation of our NewOpt is about 1.5 times *slower* than JurOpt, and has a similar performance with PaiOpt; this is a worthwhile tradeoff since key generation is run only once;
 - the encryption of our NewOpt is about 2.7 times *faster* than JurOpt and is about 7.5 times *faster* than PaiOpt;
 - the decryption of our NewOpt is about 4.1 times *faster* than JurOpt, and has a similar performance with PaiOpt.
- Refer to Sect. 6 for more details about the performance comparison.

In Table 1, we present a comparison of the algebraic structure, the provable and concrete security and the asymptotic efficiency among the four schemes. In the table, “randomness space” shows the underlying group where the randomness part of the ciphertext lies in. “assumption” shows the computational assumption under which the scheme is proved to be semantically secure. “concrete security (112-bit security)” asks the concrete parameter choices with which a 112-bit security level is achieved. “encryption” (resp. “decryption”) shows the dominating part in the encryption (resp. decryption) process. Let N denote the modulus with $N = PQ$, m the plaintext, r the randomness, and α, β factors of $(P - 1)(Q - 1)$. Let $\mathbb{Z}_N^*[+1]$ denote the set of all elements with Jacobi symbol $+1$

in $\mathbb{Z}_N^*$, $\mathbb{QR}_N$ the set of all quadratic residues in $\mathbb{Z}_N^*$, $\mathbb{QR}_N^\beta$ the set containing all β -th power of elements in $\mathbb{QR}_N$, and $\langle -1 \rangle$ the order-2 subgroup of $\mathbb{Z}_N^*$ generated by $(-1 \bmod N)$. The relations between these groups are studied in the main body and shown in Fig. 2 in Sect. 3.

Overview of our optimization. We present a high-level technical overview of our optimization NewOpt. The starting point of our optimization is a new investigation on the structure of group $\mathbb{QR}_N$, where $\mathbb{QR}_N$ denotes the set of all quadratic residues in $\mathbb{Z}_N^*$. For $N = PQ$ with properly generated primes P, Q , $\mathbb{QR}_N$ is a cyclic group of size $\phi(N)/4 = (P - 1)(Q - 1)/4$. We let α, β be divisors of $\phi(N)/4$ such that $\alpha \cdot \beta = \phi(N)/4$. In particular, we will instantiate α as a 448-bit integer which is significantly smaller than λ . Then, we decompose $\mathbb{QR}_N$ into the internal product of two subgroups $\mathbb{QR}_N^\alpha$ and $\mathbb{QR}_N^\beta$, i.e., $\mathbb{QR}_N = \mathbb{QR}_N^\alpha \otimes \mathbb{QR}_N^\beta$. Here $\mathbb{QR}_N^\alpha$ (resp. $\mathbb{QR}_N^\beta$) denotes the set containing all α -th (resp. β -th) power of elements in $\mathbb{QR}_N$. Clearly, $\mathbb{QR}_N^\alpha$ is of size β and $\mathbb{QR}_N^\beta$ is of size α . (See Subsects. 2.2 and 3.1 for more details about the structures of these subgroups. In particular, see Fig. 2 in Sect. 3.)

In our NewOpt, we carefully choose a *cyclic* subgroup $G = \mathbb{QR}_N^\beta \otimes \langle -1 \rangle$ of $\mathbb{Z}_N^*$ as the randomness space used in encryption. Here $\langle -1 \rangle$ denotes the order-2 subgroup of $\mathbb{Z}_N^*$ generated by $(-1 \bmod N)$, and the size of G is 2α . In the key generation process, we choose a generator h of G uniformly at random. The encryption algorithm is as follows

$$\begin{aligned}
 \text{Enc}_{\text{NewOpt}}(pk = (N, h), m \in \mathbb{Z}_N; r \in \mathbb{Z}_{2\alpha}) \\
 &= (1 + m \cdot N) \cdot (h^r \bmod N)^N \pmod{N^2}, \\
 &= (1 + m \cdot N) \cdot \left[(h^{\overline{2\alpha}} \bmod N^2)^r \right] \pmod{N^2},
 \end{aligned}$$

and the decryption algorithm is

$$\text{Dec}_{\text{NewOpt}}(sk = \alpha, c \in \mathbb{Z}_{N^2}^*) = \frac{(\left[c^{\overline{2\alpha}} \bmod N^2 \right] - 1)/N}{2\alpha} \pmod{N},$$

The encryption of our NewOpt involves a *fixed-base* modular exponentiation with exponent r of only roughly 448-bit, and the

decryption involves a *non-fixed-base* modular exponentiation with exponent 2α of only 448-bit. In fact, our encryption and decryption algorithms are even faster than (or have a similar performance with) those of JurOpt and PaiOpt. We refer to Sect. 6 for a detailed performance comparison among the optimized schemes.

Perspective. Intuitively, our optimization NewOpt exploits a more sophisticated decomposition of the underlying group structure (cf. Fig. 2 in Sect. 3), compared with the techniques used in the Jurik's optimization JurOpt [31] and the Paillier's optimization PaiOpt [33], thus yield substantial improvements in encryption and decryption speeds. Our optimization achieves the “best of both worlds”.

In addition to the fast encryption and decryption, our optimization is provably secure, including one-way secure and semantically secure. We refer to Sect. 4 for the formal security proofs. (There might be other ways to do optimization, but not all of them admitting a provable security as we do.)

Moreover, our optimization has a valuable impact on practice given how the Paillier scheme is being used in many MPC protocols (e.g. threshold DSA/ECDSA [27, 32]).

2 PRELIMINARIES

Security Parameters and Notations. Let $\kappa \in \mathbb{N}$ denote the security parameter throughout the paper, and let $n = n(\kappa)$, $\ell = \ell(\kappa) \in \mathbb{N}$ denote two public polynomials in κ that also determine the security level.

For $i, j \in \mathbb{N}$ with $i < j$, define $[i, j] := \{i, i+1, \dots, j\}$ and $[j] := \{1, 2, \dots, j\}$. Denote by $x \leftarrow_s \mathcal{X}$ the operation of picking an element x according to a distribution $\mathcal{X}$. If $\mathcal{X}$ is a set, then this denotes that x is sampled uniformly at random from $\mathcal{X}$. For an algorithm $\mathcal{A}$, denote by $y \leftarrow \mathcal{A}(x; r)$, or simply $y \leftarrow_s \mathcal{A}(x)$, the operation of running $\mathcal{A}$ with input x and randomness r and assigning the output to y , and by $T(\mathcal{A})$ the running time of $\mathcal{A}$. “PPT” is short for probabilistic polynomial-time. Denote by poly some polynomial function, and negl some negligible function. For a primitive XX and a security notion YY , we typically denote the advantage of a PPT adversary $\mathcal{A}$ by $\text{Adv}_{\text{XX}, \mathcal{A}}^{\text{YY}}(\kappa)$.

For an integer $N \in \mathbb{N}$, denote by $\mathbb{Z}_N$ the integral ring $\{0, 1, \dots, N-1\}$ and $\mathbb{Z}_N^*$ the group that contains all the elements in $\mathbb{Z}_N$ that are co-prime to N , i.e., $\mathbb{Z}_N^* = \{i \in \mathbb{Z}_N \mid \gcd(i, N) = 1\}$, where “gcd” stands for greatest common divisor. We will sometimes omit the modulus when it is clear from context. For integers $a, N \in \mathbb{N}$, denote by $(\frac{a}{N})$ the Jacobi symbol of a modulo N . The symbol “ $\otimes$ ” denotes internal direct product. For a group element g , denote by $\text{ord}(g)$ the order of g .

2.1 Public-Key Encryption and Semantic Security

We present the formal definition of public-key encryption (PKE) and additively homomorphic PKE. We also formalize the one-wayness and semantic security.

DEFINITION 1 (PKE). A PKE scheme $\text{PKE} = (\text{KGen}, \text{Enc}, \text{Dec})$ consists of a tuple of PPT algorithms:

- The key generation algorithm $(pk, sk) \leftarrow_s \text{KGen}(1^\kappa)$ outputs a pair of public key pk and secret key sk , where pk implicitly defines a message space $\mathcal{M}$, a randomness space $\mathcal{R}$, and a ciphertext space $\mathcal{C}$.
- The encryption algorithm $c \leftarrow \text{Enc}(pk, m; r)$ takes as input a public key pk , a message $m \in \mathcal{M}$, a randomness $r \in \mathcal{R}$, and outputs a ciphertext $c \in \mathcal{C}$.
- The decryption algorithm $m \mid \perp \leftarrow \text{Dec}(sk, c)$ takes as input a secret key sk and a ciphertext $c \in \mathcal{C}$, and outputs either a message $m \in \mathcal{M}$ or a special symbol $\perp$ indicating the failure of the decryption.

Perfect correctness of PKE requires that, for all $(pk, sk) \leftarrow_s \text{KGen}(1^\kappa)$, all messages $m \in \mathcal{M}$ and all randomnesses $r \in \mathcal{R}$, it holds that $\text{Dec}(sk, \text{Enc}(pk, m; r)) = m$.

DEFINITION 2 (ADDITIVELY HOMOMORPHIC PKE). $\text{PKE} = (\text{KGen}, \text{Enc}, \text{Dec})$ is additively homomorphic, if for any $(pk, sk) \leftarrow_s \text{KGen}(1^\kappa)$ where pk implicitly defines $\mathcal{M}, \mathcal{R}$ and $\mathcal{C}$, the following two conditions hold:

- $(\mathcal{M}, +)$ is an additive group, $(\mathcal{R}, +)$ is an (additive) group, and $(\mathcal{C}, \cdot)$ is a (multiplicative) group.
- For any $m_1, m_2 \in \mathcal{M}$, $r_1, r_2 \in \mathcal{R}$, we have $\text{Enc}(pk, m_1; r_1) \cdot \text{Enc}(pk, m_2; r_2) = \text{Enc}(pk, m_1 + m_2; r_1 + r_2)$.

DEFINITION 3 (ONE-WAYNESS FOR PKE). $\text{PKE} = (\text{KGen}, \text{Enc}, \text{Dec})$ is one-way, if for any PPT stateful adversary $\mathcal{A}$, the advantage $\text{Adv}_{\text{PKE}, \mathcal{A}}^{\text{ow}}(\kappa) := \Pr [\text{Game}_{\text{PKE}, \mathcal{A}}^{\text{ow}}(1^\kappa) = 1]$ is negligible in κ , where the security game $\text{Game}_{\text{PKE}, \mathcal{A}}^{\text{ow}}(1^\kappa)$ is specified in the left figure of Fig. 1.

$\text{Game}_{\text{PKE}, \mathcal{A}}^{\text{ow}}(1^\kappa)$:	$\text{Game}_{\text{PKE}, \mathcal{A}}^{\text{ss}}(1^\kappa)$:
$(pk, sk) \leftarrow_s \text{KGen}(1^\kappa)$. $m \leftarrow_s \mathcal{M}$. $c^* \leftarrow_s \text{Enc}(pk, m)$. $m' \leftarrow_s \mathcal{A}(pk, c^*)$. Return $(m' = m)$.	$(pk, sk) \leftarrow_s \text{KGen}(1^\kappa)$, $m \leftarrow_s \mathcal{A}(pk)$, $b \leftarrow_s \{0, 1\}$. // challenge bit If $b = 0$, $c^* \leftarrow_s \text{Enc}(pk, m)$. If $b = 1$, $R \leftarrow_s \mathcal{M}$, $c^* \leftarrow_s \text{Enc}(pk, R)$. $b' \leftarrow_s \mathcal{A}(pk, c^*)$. Return $(b' = b)$.

Figure 1: Left: the one-way security game $\text{Game}_{\text{PKE}, \mathcal{A}}^{\text{ow}}(1^\kappa)$; Right: the semantic security game $\text{Game}_{\text{PKE}, \mathcal{A}}^{\text{ss}}(1^\kappa)$.

DEFINITION 4 (SEMANTIC SECURITY FOR PKE). $\text{PKE} = (\text{KGen}, \text{Enc}, \text{Dec})$ is semantically secure (a.k.a. IND-CPA secure), if for any PPT stateful adversary $\mathcal{A}$, the advantage $\text{Adv}_{\text{PKE}, \mathcal{A}}^{\text{ss}}(\kappa) := \left| \Pr [\text{Game}_{\text{PKE}, \mathcal{A}}^{\text{ss}}(1^\kappa) = 1] - \frac{1}{2} \right|$ is negligible in κ , where the security game $\text{Game}_{\text{PKE}, \mathcal{A}}^{\text{ss}}(1^\kappa)$ is specified in the right figure of Fig. 1.

2.2 Group Structures of $\mathbb{Z}_{N^2}^*$ and $\mathbb{Z}_N^*$

The $\text{NGen}(1^\kappa)$ Algorithm. Let $\text{NGen}(1^\kappa)$ be a PPT algorithm that outputs (N, P, Q) , where $N = PQ$, and P, Q are $\frac{n(\kappa)}{2}$ -bit primes such that $P \equiv Q \equiv 3 \pmod{4}$ and $\gcd(P-1, Q-1) = 2$.

LEMMA 1 (STRUCTURE OF $\mathbb{Z}_{N^2}^*$ [17, 33]). For N output by $\text{NGen}(1^\kappa)$,

- The element $1 + N$ has order N in $\mathbb{Z}_{N^2}^*$, and for any $x \in \mathbb{Z}_N$, $(1 + N)^x = 1 + x \cdot N \pmod{N^2}$.
- Given $y = (1 + N)^x \pmod{N^2}$, it is easy to compute the discrete logarithm $x \in \mathbb{Z}_N$ by the following L algorithm:

$$x = L(y, N) := (y - 1)/N \pmod{N}.$$

- The following function is an isomorphism:

$$\begin{aligned} \psi : \mathbb{Z}_N \times \mathbb{Z}_N^* &\longrightarrow \mathbb{Z}_{N^2}^* \\ (x, y) &\longmapsto (1 + N)^x \cdot y^N \pmod{N^2} \end{aligned}$$

Structure of $\mathbb{Z}_N^*$. For N output by $\text{NGen}(1^\kappa)$, we define several important subgroups of $\mathbb{Z}_N^*$:

- Define by $\mathbb{Z}_N^*[+1]$ the set of all elements with Jacobi symbol $+1$ in $\mathbb{Z}_N^*$, i.e., $\mathbb{Z}_N^*[+1] := \{y \mid y \in \mathbb{Z}_N^*, (\frac{y}{N}) = 1\}$.
- Define by $\mathbb{QR}_N$ the set of all quadratic residues in $\mathbb{Z}_N^*$, i.e., $\mathbb{QR}_N := \{y^2 \pmod{N} \mid y \in \mathbb{Z}_N^*\}$.
- Let $\omega \in \mathbb{Z}_N^*$ be an element of order 2 and with Jacobi symbol -1 . Define by $\langle \omega \rangle$ the subgroup generated by $\omega \in \mathbb{Z}_N^*$, i.e., $\langle \omega \rangle := \{\omega \pmod{N}, 1 \pmod{N}\}$.
- Define by $\langle -1 \rangle$ the subgroup generated by $-1 \pmod{N}$ in $\mathbb{Z}_N^*$, i.e., $\langle -1 \rangle := \{-1 \pmod{N}, 1 \pmod{N}\}$.

LEMMA 2 (STRUCTURE OF $\mathbb{Z}_N^*$ [31]). For (N, P, Q) output by $\text{NGen}(1^\kappa)$, where $N = PQ$ satisfies $P \equiv Q \equiv 3 \pmod{4}$ and $\gcd(P - 1, Q - 1) = 2$, we have:

- $|\mathbb{Z}_N^*| = \phi(N) = (P - 1)(Q - 1)$.
- $\mathbb{Z}_N^*[+1]$ is a cyclic subgroup of $\mathbb{Z}_N^*$, and $|\mathbb{Z}_N^*[+1]| = \frac{1}{2} \cdot (P - 1)(Q - 1)$.
- $\mathbb{QR}_N$ is a cyclic subgroup of $\mathbb{Z}_N^*[+1]$, and $|\mathbb{QR}_N| = \frac{1}{4} \cdot (P - 1)(Q - 1)$.
- $\mathbb{Z}_N^* = \mathbb{Z}_N^*[+1] \otimes \langle \omega \rangle$, and $\mathbb{Z}_N^*[+1] = \mathbb{QR}_N \otimes \langle -1 \rangle$.

We depict the structure of $\mathbb{Z}_N^*$ in Fig. 2.

2.3 The PDL and DPDL Assumptions

DEFINITION 5 (PDL ASSUMPTION [33]). For $(N, P, Q) \leftarrow \text{sNGen}(1^\kappa)$, let α be an integer and let g be an element in $\mathbb{Z}_{N^2}^*$ of order αN . The Partial Discrete Logarithm (PDL) assumption holds w.r.t. NGen , if for any PPT adversary $\mathcal{A}$, the advantage $\text{Adv}_{\text{NGen}, \mathcal{A}}^{\text{pdl}}(\kappa) := \Pr[\mathcal{A}(N, g, c) = x] \text{ is negligible in } \kappa, \text{ where } (N, P, Q) \leftarrow \text{sNGen}(1^\kappa), r \leftarrow \mathbb{Z}_{\alpha N}, c := g^r \pmod{N^2}, x := r \pmod{N}.$

DEFINITION 6 (DPDL ASSUMPTION [33]). For $(N, P, Q) \leftarrow \text{sNGen}(1^\kappa)$, let α be an integer and let g be an element in $\mathbb{Z}_{N^2}^*$ of order αN . The Decisional Partial Discrete Logarithm (DPDL) assumption holds w.r.t. NGen , if for any PPT adversary $\mathcal{A}$, the advantage $\text{Adv}_{\text{NGen}, \mathcal{A}}^{\text{dpdl}}(\kappa) := |\Pr[\mathcal{A}(N, g, c, x) = 1] - \Pr[\mathcal{A}(N, g, c, y) = 1]|$ is negligible in κ , where $(N, P, Q) \leftarrow \text{sNGen}(1^\kappa)$, $r \leftarrow \mathbb{Z}_{\alpha N}$, $c := g^r \pmod{N^2}$, $x := r \pmod{N}$, $y \leftarrow \mathbb{Z}_N$.

2.4 Paillier's Optimization with Fast Decryption

We recall the Paillier's optimization $\text{PaiOpt} = (\text{KGen}_{\text{PaiOpt}}, \text{Enc}_{\text{PaiOpt}}, \text{Dec}_{\text{PaiOpt}})$ from [33, Section 6].

- **Key generation algorithm** $(pk, sk) \leftarrow \text{KGen}_{\text{PaiOpt}}(1^\kappa)$. It takes as input the security parameter 1^κ , which is associated with two values $n = n(\kappa)$ and $\ell = \ell(\kappa) \in \mathbb{N}$ that also determine the security level, and proceeds as follows:
 - 1) choose $\frac{n}{2}$ -bit primes P, Q and $N = PQ$;
 - 2) choose a secret ℓ -bit integer α and pick a group element $g \in \mathbb{Z}_{N^2}^*$ of order αN ;
 - 3) output $pk := (N, g)$ and $sk := \alpha$.
- **Encryption algorithm** $c \leftarrow \text{Enc}_{\text{PaiOpt}}(pk, m)$. It takes as input the public key $pk = (N, g)$ and a message $m \in \mathbb{Z}_N$, and proceeds as follows:
 - 1) pick $r \leftarrow \mathbb{Z}_N$ uniformly at random; since the order of g is αN , it is sufficient to pick r from $\mathbb{Z}_\alpha$ or approximately from $\{0, 1\}^\ell$;
 - 2) compute and output $c := g^{m+Nr} \pmod{N^2}$.
- **Decryption algorithm** $m \leftarrow \text{Dec}_{\text{PaiOpt}}(sk, c)$. It takes as input the secret key $sk = \alpha$ and a ciphertext $c \in \mathbb{Z}_{N^2}$, computes and outputs $m := \frac{L(c^\alpha \pmod{N^2, N})}{L(g^\alpha \pmod{N^2, N})} \pmod{N}$, where L is the algorithm defined in Lemma 1.

In [33], Paillier proved the one-wayness of PaiOpt under the PDL assumption (cf. Definition 5), and proved its semantic security under the DPDL assumption (cf. Definition 6). Formally, we have the following theorem.

THEOREM 1 (PDL $\Leftrightarrow$ PaiOpt is OW, DPDL $\Leftrightarrow$ PaiOpt is SS [33]). The Paillier's optimization PaiOpt is one-way if and only if the PDL assumption holds. The Paillier's optimization PaiOpt is semantically secure if and only if the DPDL assumption holds.

3 OPTIMIZED PAILLIER'S HOMOMORPHIC PKE

In this section, we optimize the Paillier's public-key encryption scheme [33]. More precisely, we investigate the structure of the subgroup $\mathbb{QR}_N$ of $\mathbb{Z}_N^*$, based on which we optimize the encryption and decryption algorithms of the Paillier scheme. Our optimization inherits the additive homomorphism.

3.1 Group Structure of $\mathbb{QR}_N$

Let $(N, P, Q) \leftarrow \text{sNGen}(1^\kappa)$. By Lemma 2, $\mathbb{QR}_N$ is a cyclic group of order $\frac{1}{4} \cdot (P - 1)(Q - 1)$. Let α, β be divisors of $\frac{1}{4} \cdot (P - 1)(Q - 1)$. We define two subgroups of $\mathbb{QR}_N$:

- Define by $\mathbb{QR}_N^\alpha$ (resp. $\mathbb{QR}_N^\beta$) the set containing all α -th (resp. β -th) power of elements in $\mathbb{QR}_N$, i.e., $\mathbb{QR}_N^\alpha := \{y^{2\alpha} \pmod{N} \mid y \in \mathbb{Z}_N^*\}$, $\mathbb{QR}_N^\beta := \{y^{2\beta} \pmod{N} \mid y \in \mathbb{Z}_N^*\}$.

LEMMA 3 (STRUCTURE OF $\mathbb{QR}_N$). For (N, P, Q) output by $\text{NGen}(1^\kappa)$ and α, β divisors of $\frac{1}{4} \cdot (P - 1)(Q - 1)$, if $\alpha \cdot \beta = \frac{1}{4} \cdot (P - 1)(Q - 1)$ and $\gcd(\alpha, \beta) = 1$, it holds that

- $\mathbb{QR}_N^\alpha$ and $\mathbb{QR}_N^\beta$ are cyclic subgroups of $\mathbb{QR}_N$, $|\mathbb{QR}_N^\alpha| = \beta$, and $|\mathbb{QR}_N^\beta| = \alpha$.

^{*}In [33, Section 6], Paillier did not specify how to choose α and g concretely.

$$- \mathbb{QR}_N = \mathbb{QR}_N^\alpha \otimes \mathbb{QR}_N^\beta.$$

We depict the structure of $\mathbb{QR}_N$ in Fig. 2.

PROOF. Clearly, $\mathbb{QR}_N^\alpha$ and $\mathbb{QR}_N^\beta$ are subgroups of $\mathbb{QR}_N$. By Lemma 2, $\mathbb{QR}_N$ is cyclic and $|\mathbb{QR}_N| = \frac{1}{4} \cdot (P-1)(Q-1) = \alpha \cdot \beta$. Thus, $\mathbb{QR}_N^\alpha$ and $\mathbb{QR}_N^\beta$ are also cyclic, the order of $\mathbb{QR}_N^\alpha$ is β and the order of $\mathbb{QR}_N^\beta$ is α .

By $\gcd(\alpha, \beta) = 1$, it holds that $\mathbb{QR}_N^\alpha \cap \mathbb{QR}_N^\beta = \{1 \pmod{N}\}$. Consequently, $\mathbb{QR}_N^\alpha \otimes \mathbb{QR}_N^\beta \subseteq \mathbb{QR}_N$. Together with $|\mathbb{QR}_N^\alpha \otimes \mathbb{QR}_N^\beta| = \beta \cdot \alpha = \frac{1}{4} \cdot (P-1)(Q-1) = |\mathbb{QR}_N|$, we have that $\mathbb{QR}_N^\alpha \otimes \mathbb{QR}_N^\beta = \mathbb{QR}_N$. $\square$

Figure 2: The structures of $\mathbb{Z}_N^*$ and $\mathbb{QR}_N$, where the latter is highlighted with dotted box. By Lemma 2 and Lemma 3, $\mathbb{Z}_N^*[+1]$, $\mathbb{QR}_N$, $\mathbb{QR}_N^\alpha$, $\mathbb{QR}_N^\beta$ are cyclic subgroups of $\mathbb{Z}_N^*$. In our optimized Paillier scheme in Subsect. 3.2, we choose $G = \mathbb{QR}_N^\beta \otimes \langle -1 \rangle$, a specific cyclic subgroup of $\mathbb{Z}_N^*$, as the randomness space during encryption. We highlight G in gray. By Lemma 3, the order of G is 2α .

3.2 Optimized Paillier Scheme

The $\text{NGen}(1^\kappa)$ Algorithm, Revisited. From this subsection on, we require the $\text{NGen}(1^\kappa)$ algorithm to output another pair of (p, q) besides (N, P, Q) . Formally, let $\text{NGen}(1^\kappa)$ be a PPT algorithm that outputs (N, P, Q, p, q) , where $N = PQ$, p, q are $\frac{\ell(\kappa)}{2}$ -bit odd primes and P, Q are $\frac{n(\kappa)}{2}$ -bit primes such that

$$p \mid (P-1), q \mid (Q-1), P \equiv Q \equiv 3 \pmod{4}, \quad (2)$$

$$\gcd(P-1, Q-1) = 2, \gcd(pq, (P-1)(Q-1)/(4pq)) = 1.$$

One way to implement $\text{NGen}(1^\kappa)$ is:

- (1) choose $\frac{\ell(\kappa)}{2}$ -bit odd primes p, q and $(\frac{n(\kappa)-\ell(\kappa)}{2} - 1)$ -bit odd integers p', q' randomly;
- (2) set $P := 2pp' + 1$ and $Q := 2qq' + 1$;
- (3) test whether P and Q are primes and whether p, q, p', q' are co-prime with each other;
- (4) if the test passes, output $(N = PQ, P, Q, p, q)$; otherwise go back to step (1).

The Optimized Paillier Scheme. We present our $\text{NewOpt} = (\text{KGen}_{\text{NewOpt}}, \text{Enc}_{\text{NewOpt}}, \text{Dec}_{\text{NewOpt}})$ with message space $\mathcal{M} = \mathbb{Z}_N$ as follows.

- **Key generation algorithm** $(pk, sk) \leftarrow \text{KGen}_{\text{NewOpt}}(1^\kappa)$.

It takes as input the security parameter 1^κ , which is associated with two values $n = n(\kappa)$ and $\ell = \ell(\kappa) \in \mathbb{N}$ that also determine the security level, and proceeds as follows:

- 1) invoke $(N, P, Q, p, q) \leftarrow \text{NGen}(1^\kappa)$ such that (2) holds;
- 2) compute $\alpha := pq$ and $\beta := (P-1)(Q-1)/(4pq)$;
- 3) pick $y \leftarrow \mathbb{Z}_N^*$ uniformly at random, and compute $h := -y^{2\beta} \pmod{N}$;
- 4) output $pk := (N, h)$ and $sk := \alpha$.

Remark. Clearly, $\alpha \cdot \beta = (P-1)(Q-1)/4$ and $\gcd(\alpha, \beta) = 1$, thus by Lemma 3, $\mathbb{QR}_N^\beta$ is a cyclic subgroup of $\mathbb{Z}_N^*$ of order α . Then $h = -y^{2\beta} \pmod{N}$ is a generator of the subgroup $G = \mathbb{QR}_N^\beta \otimes \langle -1 \rangle$ (see Fig. 2), whose order is 2α .

We stress that the parameter g in PaiOpt 's public key is an element in $\mathbb{Z}_{N^2}^*$ and has order αN (cf. Subsect. 2.4), while the parameter h in our NewOpt 's public key is an element in $\mathbb{Z}_N^*$ and has order 2α . Moreover, our choice of parameter h is concrete, i.e., $h := -y^{2\beta} \pmod{N}$ for a uniformly chosen $y \leftarrow \mathbb{Z}_N^*$.[†]

- **Encryption algorithm** $c \leftarrow \text{Enc}_{\text{NewOpt}}(pk, m)$.

It takes as input the public key $pk = (N, h)$ and a message $m \in \mathbb{Z}_N$, and proceeds as follows:

- 1) pick $r \leftarrow \{0, 1\}^\ell$ uniformly at random, and regard r as an integer in $\mathbb{Z}$ naturally;
- 2) compute and output

$$c := (1 + N)^m \cdot (h^r \pmod{N})^N \pmod{N^2}. \quad (3)$$

Remark. Here, the term $(h^r \pmod{N})$ is statistically close to the uniform distribution over the subgroup $G = \mathbb{QR}_N^\beta \otimes \langle -1 \rangle$ (see Fig. 2).

- **Decryption algorithm** $m \leftarrow \text{Dec}_{\text{NewOpt}}(sk, c)$.

It takes as input the secret key $sk = \alpha$ and a ciphertext $c \in \mathbb{Z}_{N^2}$, computes and outputs

$$m := \text{L}(c^{2\alpha} \pmod{N^2}, N) \cdot (2\alpha)^{-1} \pmod{N}, \quad (4)$$

where L is the algorithm defined in Lemma 1.

The following theorems show the correctness and the additive homomorphism of our optimized Paillier scheme. We omit their proofs here and postpone them to Appendix A.

THEOREM 2 (CORRECTNESS). For all possible messages $m \in \mathbb{Z}_N$ and randomnesses $r \in \{0, 1\}^\ell$, the decryption (4) always undoes the encryption (3).

THEOREM 3 (ADDITIVE HOMOMORPHISM). For all possible messages $m_1, m_2 \in \mathbb{Z}_N$ and randomnesses $r_1, r_2 \in \{0, 1\}^\ell$, it holds that $\text{Enc}_{\text{NewOpt}}(pk, m_1; r_1) \cdot \text{Enc}_{\text{NewOpt}}(pk, m_2; r_2) = \text{Enc}_{\text{NewOpt}}(pk, m_1 + m_2; r_1 + r_2)$.

[†]Jumping ahead, one can also substantiate the choice of parameter h over the use of parameter g . See (5) and the analysis there in Subsect. 4.1 for more details.

4 ONE-WAYNESS AND SEMANTIC SECURITY

In this section, we establish the one-wayness and semantic security of our optimized Paillier scheme NewOpt (cf. Subsect. 3.2) upon the security of the Paillier's optimization PaiOpt [33] (cf. Subsect. 2.4).

4.1 One-Wayness of Our NewOpt

In this subsection, we show that the one-wayness of our optimized Paillier scheme NewOpt in Subsect. 3.2 is as strong as that of the Paillier's optimization PaiOpt.

THEOREM 4 (PaiOpt is OW $\Rightarrow$ NewOpt is OW). *If the Paillier's optimization PaiOpt in [33] is one-way, so is our optimized Paillier scheme NewOpt in Subsect. 3.2. Concretely, for any adversary $\mathcal{A}$ against the one-wayness of NewOpt, there exists an adversary $\mathcal{B}$ against the one-wayness of PaiOpt, such that $T(\mathcal{B}) \approx T(\mathcal{A})$ and $\text{Adv}_{\text{NewOpt}, \mathcal{A}}^{\text{ow}}(\kappa) \leq \text{Adv}_{\text{PaiOpt}, \mathcal{B}}^{\text{ow}}(\kappa) + 2^{-\Omega(\kappa)}$.*

PROOF. We construct $\mathcal{B}$ against the one-wayness of PaiOpt. Given a challenge (pk, c^*) , where $pk = (N, g)$ and $c^* \leftarrow \text{Enc}_{\text{PaiOpt}}(pk, m; r)$ with $m \leftarrow \mathbb{Z}_N$ and $r \leftarrow \{0, 1\}^\ell$, $\mathcal{B}$ wants to recover the message m from (pk, c^*) . $\mathcal{B}$ will convert (pk, c^*) of scheme PaiOpt to a related pair of public key and ciphertext (pk', c'^*) of scheme NewOpt, and employ the power of $\mathcal{A}$ to recover m .

We detail how $\mathcal{B}$ simulates (pk', c'^*) for $\mathcal{A}$.

- **Simulating the public key.** Given $pk = (N, g)$, $\mathcal{B}$ simulates $pk' = (N, h)$ as follows:

$$pk' := (N, h), \text{ where } h := -g \pmod{N}. \quad (5)$$

Next, we analyze the distribution of pk' . For $pk = (N, g)$ generated by $\text{KGen}_{\text{PaiOpt}}(1^\kappa)$, we have $g = (1 + N) \cdot (z^{2\beta})^N \pmod{N^2}$ with $z \leftarrow \mathbb{Z}_N^*$. Thus,

$$\begin{aligned} h &= -g \pmod{N} = -1 \cdot (z^{2\beta})^N \pmod{N} \\ &= -1 \cdot (z^N)^{2\beta} \pmod{N}. \end{aligned}$$

By implicitly setting $y := z^N \pmod{N}$, which is uniformly distributed over $\mathbb{Z}_N^*$, it holds that $h = -y^{2\beta} \pmod{N}$. Consequently, $pk' = (N, h)$ constitutes a valid public key for NewOpt.

- **Simulating the ciphertext.** Given c^* , $\mathcal{B}$ simulates c'^* as follows:

$$c'^* := (c^*)^2 \cdot (h^{r'} \pmod{N})^N \pmod{N^2}, \text{ where } r' \leftarrow \{0, 1\}^\ell. \quad (6)$$

Next, we analyze the distribution of c'^* . For $c^* \leftarrow \text{Enc}_{\text{PaiOpt}}(pk, m; r)$, we have $c^* = g^{m+Nr} \pmod{N^2}$. Let $\delta := N^{-1} \pmod{\phi(N)}$, which might not be efficiently computable. By $y = z^N \pmod{N}$, it follows that $z = y^\delta \pmod{N}$.

Consequently, we have

$$\begin{aligned} c'^* &= (g^{m+Nr})^2 \cdot (h^{r'} \pmod{N})^N \\ &= \psi(1, z^{2\beta})^{2(m+Nr)} \cdot \psi(0, h^{r'}) \end{aligned} \quad (7)$$

$$= \psi(2m, z^{4\beta(m+Nr)} \cdot h^{r'}) \quad (8)$$

$$= \psi(2m, (y^\delta)^{4\beta(m+Nr)} \cdot h^{r'}) \quad (9)$$

$$= \psi(2m, y^{4\beta(\delta m+r)} \cdot h^{r'}) \quad (10)$$

$$= \psi(2m, (-h)^{2(\delta m+r)} \cdot h^{r'}) \quad (11)$$

$$= \psi(2m, h^{2(\delta m+r)+r'})$$

$$\approx \psi(2m, h^{r''}) \text{ with } r'' \leftarrow \{0, 1\}^\ell \quad (12)$$

$$= \text{Enc}_{\text{NewOpt}}(pk', 2m; r'') \text{ with } r'' \leftarrow \{0, 1\}^\ell,$$

where (7) follows from $g = (1 + N) \cdot (z^{2\beta})^N \pmod{N^2} = \psi(1, z^{2\beta})$, (8) from the fact that ψ is an isomorphism, (9) from $z = y^\delta \pmod{N}$, (10) from $\delta = N^{-1} \pmod{\phi(N)}$, (11) from $h = -y^{2\beta} \pmod{N}$, and (12) holds due to the fact that $2(\delta m+r)+r' \pmod{\text{ord}(h)}$ is statistically close to the uniform distribution over $\{0, 1\}^\ell$ by the randomness of r' .

Therefore, c'^* is a fresh ciphertext of $2m$ under the public key $pk' = (N, h)$ using $\text{Enc}_{\text{NewOpt}}(pk', 2m; r'')$.

$\mathcal{B}$ simulates (pk', c'^*) for $\mathcal{A}$ as described above. Finally, $\mathcal{B}$ receives m' from $\mathcal{A}$, and $\mathcal{B}$ outputs $m' \cdot 2^{-1} \pmod{N}$ to its own challenger. According to the above analysis, $\mathcal{B}$ perfectly simulates the one-way security game of NewOpt for $\mathcal{A}$, except with an exponentially small statistical distance $2^{-\Omega(\kappa)}$ introduced by (12), and $\mathcal{B}$ succeeds (i.e., $m' \cdot 2^{-1} = m$) as long as $\mathcal{A}$ does (i.e., $m' = 2m$). It follows that $\text{Adv}_{\text{PaiOpt}, \mathcal{B}}^{\text{ow}}(\kappa) \geq \text{Adv}_{\text{NewOpt}, \mathcal{A}}^{\text{ow}}(\kappa) - 2^{-\Omega(\kappa)}$. This completes the proof of Theorem 4. $\square$

By combining Theorems 1 and 4 together, we have the following corollary.

COROLLARY 1 (ONE-WAYNESS OF OUR NewOpt). *Our optimized Paillier scheme NewOpt in Subsect. 3.2 is one-way under the PDL assumption.*

4.2 Semantic Security of Our NewOpt

In this subsection, we show that the semantic security of our optimized Paillier scheme NewOpt in Subsect. 3.2 is as strong as that of the Paillier's optimization PaiOpt.

THEOREM 5 (PaiOpt is SS $\Rightarrow$ NewOpt is SS). *If the Paillier's optimization PaiOpt in [33] is semantically secure, so is our optimized Paillier scheme NewOpt in Subsect. 3.2. Concretely, for any adversary $\mathcal{A}$ against the semantic security of NewOpt, there exists an adversary $\mathcal{B}$ against the semantic security of PaiOpt, such that $T(\mathcal{B}) \approx T(\mathcal{A})$ and $\text{Adv}_{\text{NewOpt}, \mathcal{A}}^{\text{ss}}(\kappa) \leq \text{Adv}_{\text{PaiOpt}, \mathcal{B}}^{\text{ss}}(\kappa) + 2^{-\Omega(\kappa)}$.*

PROOF. We construct $\mathcal{B}$ against the semantic security of PaiOpt. Given a public key $pk = (N, g)$ of scheme PaiOpt, $\mathcal{B}$ will convert it to a related public key pk' of scheme NewOpt and send pk' to $\mathcal{A}$. After receiving m from $\mathcal{A}$, $\mathcal{B}$ forwards m to its own challenger and obtains a challenge ciphertext c^* . $\mathcal{B}$ wants to distinguish the following two cases in the semantic security game $\text{Game}_{\text{PaiOpt}, \mathcal{B}}^{\text{ss}}(1^\kappa)$ (cf. Fig. 1):

- Case $b = 0$: $c^* \leftarrow \text{Enc}_{\text{PaiOpt}}(pk, m; r)$, which is an encryption of m .
- Case $b = 1$: $c^* \leftarrow \text{Enc}_{\text{PaiOpt}}(pk, R; r)$, which is an encryption of a random message $R \leftarrow \mathbb{Z}_N$.

$\mathcal{B}$ will convert c^* of scheme PaiOpt to a related ciphertext c'^* of scheme NewOpt, and employ the power of $\mathcal{A}$ to distinguish which case it faces.

We detail how $\mathcal{B}$ simulates the public key pk' and ciphertext c'^* for $\mathcal{A}$.

- **Simulating the public key.** The simulation of pk' is the same as that in the proof of Theorem 4, i.e., according to (5). More precisely, given $pk = (N, g)$ of scheme PaiOpt, $\mathcal{B}$ computes $h := -g \pmod{N}$ and sets $pk' := (N, h)$. Following the same analysis in the proof of Theorem 4, it implicitly sets $h = -y^{2\beta} \pmod{N}$ with $y := z^N \pmod{N}$, thus $pk' = (N, h)$ constitutes a valid public key for NewOpt.
- **Simulating the ciphertext.** Given c^* , $\mathcal{B}$ simulates c'^* as follows:

$$c'^* := (c^*)^2 \cdot (1 + N)^{-m} \cdot (h^{r'} \pmod{N})^N \pmod{N^2}, \quad (13)$$

where $r' \leftarrow \{0, 1\}^\ell$.

Next, we analyze the distribution of c'^* . For $c^* \leftarrow \text{Enc}_{\text{PaiOpt}}(pk, m; r)$ (in the case of $b = 0$) or $c^* \leftarrow \text{Enc}_{\text{PaiOpt}}(pk, R; r)$ (in the case of $b = 1$), we have $c^* = g^{b(R-m)+m+Nr} \pmod{N^2}$. Let $\delta := N^{-1} \pmod{\phi(N)}$, which might not be efficiently computable. By $y = z^N \pmod{N}$, it follows that $z = y^\delta \pmod{N}$. Consequently, we have

$$\begin{aligned} c'^* &= (g^{b(R-m)+m+Nr})^2 \cdot (1 + N)^{-m} \cdot (h^{r'} \pmod{N})^N \\ &= \psi(1, z^{2\beta})^{2(b(R-m)+m+Nr)} \cdot \psi(-m, h^{r'}) \end{aligned} \quad (14)$$

$$= \psi(2b(R-m) + m, z^{4\beta(b(R-m)+m+Nr)} \cdot h^{r'}) \quad (15)$$

$$= \psi(2b(R-m) + m, (y^\delta)^{4\beta(b(R-m)+m+Nr)} \cdot h^{r'}) \quad (16)$$

$$= \psi(2b(R-m) + m, y^{4\beta(\delta b(R-m)+\delta m+r)} \cdot h^{r'}) \quad (17)$$

$$= \psi(2b(R-m) + m, (-h)^{2(\delta b(R-m)+\delta m+r)} \cdot h^{r'}) \quad (18)$$

$$= \psi(2b(R-m) + m, h^{2(\delta b(R-m)+\delta m+r)+r'}) \quad (19)$$

$$\stackrel{s}{\approx} \psi(2b(R-m) + m, h^{r''}) \text{ with } r'' \leftarrow \{0, 1\}^\ell$$

$$= \text{Enc}_{\text{NewOpt}}(pk', 2b(R-m) + m; r'') \text{ with } r'' \leftarrow \{0, 1\}^\ell,$$

where (14) follows from $g = (1 + N) \cdot (z^{2\beta})^N \pmod{N^2} = \psi(1, z^{2\beta})$, (15) from the fact that ψ is an isomorphism, (16) from $z = y^\delta \pmod{N}$, (17) from $\delta = N^{-1} \pmod{\phi(N)}$, (18) from $h = -y^{2\beta} \pmod{N}$, and (19) holds due to the fact that $2(\delta b(R-m) + \delta m + r) + r' \pmod{\text{ord}(h)}$ is statistically close to the uniform distribution over $\{0, 1\}^\ell$ by the randomness of r' .

Therefore, in the case of $b = 0$, c'^* is a fresh ciphertext of m under the public key $pk' = (N, h)$ using $\text{Enc}_{\text{NewOpt}}(pk', m; r'')$, while in the case of $b = 1$, c'^* is a fresh ciphertext of $R' := 2R - m$ under the public key $pk' = (N, h)$ using $\text{Enc}_{\text{NewOpt}}(pk', 2R - m; r'')$, where R' is a random message in $\mathbb{Z}_N$.

$\mathcal{B}$ simulates the public key pk' and the challenger ciphertext c'^* for $\mathcal{A}$ as described above and outputs whatever $\mathcal{A}$ outputs. According to the above analysis,

- if c^* is an encryption of m for PaiOpt (i.e., in the case of $b = 0$), c'^* is an encryption of m for NewOpt;
- if c^* is an encryption of R for PaiOpt (i.e., in the case of $b = 1$), c'^* is an encryption of $R' = 2R - m$ for NewOpt.

Therefore, $\mathcal{B}$ perfectly simulates the semantic security game of NewOpt for $\mathcal{A}$, except with an exponentially small statistical distance $2^{-\Omega(\kappa)}$ introduced by (19), and $\mathcal{B}$ succeeds as long as $\mathcal{A}$ does. It follows that $\text{Adv}_{\text{PaiOpt}, \mathcal{B}}^{ss}(\kappa) \geq \text{Adv}_{\text{NewOpt}, \mathcal{A}}^{ss}(\kappa) - 2^{-\Omega(\kappa)}$. This completes the proof of Theorem 5. $\square$

By combining Theorems 1 and 5 together, we have the following corollary.

COROLLARY 2 (SEMANTIC SECURITY OF OUR NewOpt). *Our optimized Paillier scheme NewOpt in Subsect. 3.2 is semantically secure under the DPDL assumption.*

5 SECURITY ANALYSIS AND CONCRETE CHOICES OF PARAMETERS

In this section, we analyze the security of our optimized Paillier scheme NewOpt in Subsect. 3.2 under several known methods.

Integer Factorization Algorithms. A direct way to defeat the security of our NewOpt is by factorizing the modulus N . In our scheme, N is an n -bit integer and is a product of two $\frac{n}{2}$ -bit primes P, Q , which satisfy

$$P \equiv Q \equiv 3 \pmod{4}, \gcd(P-1, Q-1) = 2, \text{ and } p \mid (P-1), q \mid (Q-1)$$

with $\frac{\ell}{2}$ -bit primes p, q . According to the NIST technical report [1], a choice of 2048-bit RSA modulus N is estimated to provide a 112-bit security, and a choice of 3072-bit RSA modulus N is estimated to provide a 128-bit security.

The Pollard's $P-1$ algorithm [34] is a number theoretic integer factorization algorithm. When $P-1$ is B -powersmooth, i.e., all prime powers p^v dividing $P-1$ are upper bounded by some $B \in \mathbb{N}$, the algorithm can be used to find the factor P of N in $\Omega(B \cdot \log N / \log B)$ steps. In our scheme, N has only two factors P, Q . Moreover, $P-1$ (resp. $Q-1$) has an $\frac{\ell}{2}$ -bit prime factor p (resp. q), thus it is B -powersmooth for $B \geq 2^{\ell/2}$. Consequently, the complexity of the Pollard's $P-1$ algorithm is $\Omega(B \cdot \log N / \log B) \geq \Omega(2^{\ell/2})$. This suggests that a choice of $\ell \geq 224$ would provide at least 112-bit security and a choice of $\ell \geq 256$ would provide at least 128-bit security. To gain more confidence on the security, one could generate $\frac{n}{2}$ -bit primes P, Q by setting $P = 2pp' + 1$ and $Q = 2qq' + 1$ for $\frac{\ell}{2}$ -bit primes p, q and $(\frac{n}{2} - \frac{\ell}{2} - 1)$ -bit primes p', q' .

Specific Integer Factorization Algorithm on Small Secret CRT-Exponents. Actually, one could factorize the modulus N more efficiently with the help of h contained in the public key. Note that, in our NewOpt, $pk = (N, h)$ where h is a generator of the subgroup $G = \mathbb{QR}_N^\beta \otimes \langle -1 \rangle$ and has order 2α (see Subsect. 3.2 and Fig. 2). Therefore, it holds that $h^{2\alpha} = 1 \pmod{N}$. Here α is an ℓ -bit

long secret integer, $\alpha = pq$ with p and q of bit-length $\ell/2$, where p divides $(P-1)$ and q divides $(Q-1)$. Then by the Chinese Remainder Theorem (CRT), it holds that

$$h^{2p} = 1 \pmod{P}.$$

Obviously P and p are unknown, but one could recover P and p in $\tilde{O}(\sqrt{p}) = \tilde{O}(2^{\ell/4})$ operations by adapting the classical attack on small secret-exponent in RSA-CRT [6].

More precisely, we have

$$\gcd(h^{2p} - 1 \pmod{N}, N) = P.$$

We have to evaluate all these gcd's in time $\tilde{O}(\sqrt{p})$. If one writes p as $p = p_0 + T \cdot p_1$ with $T = 2^{\ell/4}$ and p_0 and p_1 of similar size ($2^{\ell/4}$), we can evaluate $\gcd(R(h^{2p_0}) \pmod{N}, N)$ for all possible p_0 (less than $2^{\ell/4}$) where $R(\cdot)$ is the polynomial

$$R(X) = (X-1) \cdot (X \cdot (h^2)^T - 1) \cdot (X \cdot (h^2)^{2T} - 1) \cdots (X \cdot (h^2)^{2^{\ell/4}T} - 1) \pmod{N}$$

of degree $2^{\ell/4}$. Using classical multi-evaluations techniques of univariate polynomials, this gives an algorithm with complexity $\tilde{O}(2^{\ell/4})$. This suggests that

- To provide 112-bit security, ℓ should be at least 448-bit.
- To provide 128-bit security, ℓ should be at least 512-bit.

The Baby-Step Giant-Step Algorithm. Another way to attack our NewOpt is by guessing the randomness r used in the encryption and then recovering the plaintext m from the ciphertext c . This will take $2^{\ell-1}$ tries for an ℓ -bit r on average. We can use the baby-step giant-step algorithm (BSGS) [37] to improve the result. BSGS is a time-memory trade-off algorithm for solving the discrete logarithm problem (DLP) over a generic cyclic group.

Given a ciphertext $c = (1 + N)^m \cdot (h^r \pmod{N})^N \pmod{N^2}$, one could first reduce it modulo N and obtain

$$c \pmod{N} = (h^r \pmod{N})^N \pmod{N} = (h^N \pmod{N})^r \pmod{N}.$$

Then the randomness r is the discrete logarithm of $(c \pmod{N})$ w.r.t. $(h^N \pmod{N})$. The base $(h^N \pmod{N})$ is a generator of the subgroup $G = \mathbb{QR}_N^\beta \otimes \langle -1 \rangle$ whose order is 2α (see Subsect. 3.2 and Fig. 2). We apply BSGS on this problem. The complexity of BSGS over a cyclic group of size 2α is $\Omega(2\alpha)$, which is about $\Omega(2^{\ell/2})$ for an ℓ -bit α .

Similarly, one could use the BSGS algorithm to recover the secret key $sk = \alpha$ from the public key $pk = (N, h)$. In our optimized scheme, h is a generator of the subgroup $G = \mathbb{QR}_N^\beta \otimes \langle -1 \rangle$ whose order is 2α (see Subsect. 3.2 and Fig. 2). Therefore, it holds that

$$h^{2\alpha} = 1 \pmod{N}.$$

By using the BSGS algorithm to recover the discrete logarithm 2α , one could obtain the secret key α . The complexity is also $\Omega(\sqrt{2\alpha})$, which is about $\Omega(2^{\ell/2})$ for an ℓ -bit α .

Consequently,

- For 2048-bit modulus N that provides security strength of 112 bits, ℓ should be at least 224-bit.
- For 3072-bit modulus N that provide security strength of 128 bits, ℓ should be at least 256-bit.

The Coppersmith's Method. From $c \pmod{N} = (h^r \pmod{N})^N \pmod{N}$, it seems that we can consider the Coppersmith's method for finding small roots of polynomials [15].

Table 2: Concrete choices of parameters n, ℓ with respect to the security parameter κ . Here n is the bit-length of the modulus N , and ℓ is the bit-length of the underlying subgroup $G = \mathbb{QR}_N^\beta \otimes \langle -1, 1 \rangle$ which is used as the randomness space in the encryption of our optimized Paillier scheme.

security level κ (bit)	bit-length n of the modulus	bit-length ℓ of the subgroup
112	2048	448
128	3072	512
192	7680	768

But the degree N of the polynomial is exponentially large that we can rule out the Coppersmith's method totally.

Concrete Choices of Parameters. According to the above analysis, we list our concrete parameter choices for the security parameter κ , the bit-length $n = n(\kappa)$ of the modulus N , and the bit-length $\ell = \ell(\kappa)$ of the underlying subgroup $G = \mathbb{QR}_N^\beta \otimes \langle -1 \rangle$, in Table 2. Note that the suggestion on n is due to the NIST technical report [1].

6 PERFORMANCE EVALUATION

In this section, we present a thorough experimental evaluation of our optimized Paillier scheme NewOpt, and compare it with the Jurik's optimization JurOpt [31] and the Paillier's optimization PaiOpt (i.e., the Scheme 3 in [33]). We also present the experimental evaluation of the basic Paillier scheme [33] as a baseline.

We implemented the four schemes on a 2018 MacBook Pro (Intel Core i7 at 2.7 GHz, 16GB RAM at 2.133 GHz) running macOS 10.4.5, using C language. The implementations were single-threaded. We used OpenSSL 1.1.1a open-source library [2] for the generation of prime numbers and for the modulo operations over $\mathbb{Z}_N$ and $\mathbb{Z}_{N^2}$.

We implemented the four schemes with parameters $n = 2048$ (which is the bit-length of N) and $\ell = 448$ (which is the bit-length of α in our optimized scheme and in the Paillier's optimization). By our analysis in Sect. 5 (cf. Table 2), such parameter choices will provide a 112-bit security level.

For each scheme, we implemented the key generation algorithm KGen, the encryption algorithm Enc and the decryption algorithm Dec, respectively. The experimental results are summarized in Table 3. Specifically, for Enc where a fixed-base modular exponentiation is involved, we give two flavors of implementations, one is "without any further speed-up" which means a straightforward implementation, the other is "speed-up using pre-computation" which means implementation by pre-computing powers of a fixed base which constitutes a pre-computed table. For the latter, we use pre-computed tables of two different sizes, 16 MB and 100 MB, respectively. We note that the Enc of the basic Paillier scheme involves a non-fixed-based modular exponentiation, thus does not support the "speed-up using pre-computation" implementation. For Dec, we also present two flavors of implementations, one is "without any further speed-up" which means a straightforward implementation, the other is "speed-up using CRT" which means implementation by applying the Chinese Remainder Theorem

(CRT) [20, 33].

Performance Evaluation of KGen. For each scheme, we tested the KGen algorithm 50 times independently, and show the average running time in Table 3. We also depict the experiment results in Fig. 3 in Appendix B. The experiments show that the KGen algorithm of our NewOpt is about 1.5 times slower than JurOpt, and has a similar performance with PaiOpt.

Performance Evaluation of Enc and Speed-up by Pre-computation. For each scheme, we tested the Enc algorithm 5000 times independently for encrypting 2048-bit messages, and show the average running time in Table 3. We also depict the experiment results in the left figure of Fig. 4 in Appendix B.

The Enc algorithm of the basic Paillier scheme involves a non-fixed-base modular exponentiation, while the Enc algorithm of the three optimized schemes involves only *fixed-base* modular exponentiations. In the latter case, we can accelerate the Enc algorithm by pre-computation. For example, in order to compute $h = g^x$ for a fixed base g , we can pre-compute powers of the fixed base g and turn the expensive modular exponentiations into “cheap” modular multiplications. To be more specific, assuming that x is an ℓ -bit integer, we can divide x into $\lceil \ell/b \rceil$ blocks of b -bit as follows: $x = \sum_{i=1}^{\lceil \ell/b \rceil} x_i \cdot 2^{ib}$, where $x_i \in \{0, 1\}^b$ is the i -th block (the last block $x_{\lceil \ell/b \rceil}$ may be less than b -bit). Then

$$h = g^x = g^{\sum_{i=1}^{\lceil \ell/b \rceil} x_i 2^{ib}} = \prod_{i=1}^{\lceil \ell/b \rceil} (g^{2^{ib}})^{x_i}. \quad (20)$$

We could pre-compute a table consisting of $\{(i, j, (g^{2^{ib}})^j) \mid i \in [1, \lceil \ell/b \rceil], j \in [0, 2^b - 1]\}$. With this pre-computed table, $(\lceil \ell/b \rceil - 1)$ modular multiplications suffices for the fixed-base modular exponentiation $h = g^x$ according to (20). The pre-computed table consists of $\lceil \ell/b \rceil \cdot 2^b$ elements in $\mathbb{Z}_{N^2}$, whose size is $\lceil \ell/b \rceil \cdot 2^b \cdot (2n)$ bits.

Besides the straightforward implementation, we also gave implementation of the Enc algorithm by pre-computation. More precisely, for each of the three optimized schemes (JurOpt, PaiOpt and our NewOpt), we also implemented the Enc algorithm using pre-computed tables of two different sizes, 16 MB and 100 MB, respectively.

Our experiments show that the generation of a same-sized (16 MB or 100 MB) pre-computed table costs a similar time for the three optimized schemes. With a same-sized (16 MB or 100 MB) pre-computed table, the Enc algorithm of our NewOpt is about 2.7 times faster than JurOpt, and is about 7.5 times faster than PaiOpt.

Performance Evaluation of Dec and Speed-up by CRT. For each scheme, we tested the Dec algorithm 5000 times independently, and show the average running time in Table 3. We also depict the experiment results in the right figure of Fig. 4 in Appendix B.

The Dec algorithm of all the four schemes involves a non-fixed-base modular exponentiation with modulus N^2 , where $N = PQ$. If we put the factors P and Q into the secret key, we can employ the CRT to accelerate the Dec algorithm efficiently. See [33] for more details. Besides the straightforward implementation, we also implemented Dec using CRT.

The experiments show that, if all of the four schemes employ CRT in the implementation, the Dec algorithm of our NewOpt is

about 4.1 times faster than JurOpt, and has a similar performance with PaiOpt.

7 CONCLUSION

In this paper, we propose a new optimization for the widely-used Paillier’s public-key encryption. Our optimized scheme greatly improves the performance of both encryption and decryption algorithms and retains the useful additively homomorphic property. Comprehensive experiments show that, under a same security level and with a same-sized pre-computed table, the encryption algorithm of our optimization is about 7.5 times faster than that of the Paillier’s optimization (i.e., the Scheme 3 in [33]) and is about 2.7 times faster than that of the Jurik’s optimization [31]; the decryption algorithm of our scheme is about 4.1 times faster than that of the Jurik’s optimization and has a similar performance with that of the Paillier’s optimization. Moreover, we prove that our optimization is at least as secure as the Paillier’s optimization.

REFERENCES

- [1] 2016. NIST: Recommendation for key management, Special Publication 800-57 Part 1 Rev.4 (2016).
- [2] 2019. OpenSSL: OpenSSL 1.1.1a open-source library. (2019). <https://github.com/openssl/openssl/>
- [3] Michael Backes, Dario Fiore, and Paphael M. Reischuk. 2013. Verifiable delegation of computation on outsourced data. In: *Sadeghi, A., Gligor, V.D., Yung, M. (eds.) CCS 2013*, pp. 863–874. ACM (2013).
- [4] J.D.C. Benaloh. 1987. Verifiable secret-ballot elections. *PhD Thesis* (1987).
- [5] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. 2011. Semi-homomorphic encryption and multiparty computation. In: *Paterson, K.G. (ed.) EUROCRYPT 2011*, LNCS, vol. 6632, pp. 169–188. Springer (2011).
- [6] Daniel Bleichenbacher and Alexander May. 2006. New attacks on RSA with small secret CRT-exponents. In: *Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006*, LNCS, vol. 3958, pp. 1–13. Springer (2006).
- [7] Dan Boneh and David ManDell Freeman. 2011. Homomorphic signatures for polynomial functions. In: *Paterson, K.G. (ed.) EUROCRYPT 2011*, LNCS, vol. 6632, pp. 149–168. Springer (2011).
- [8] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2012. (Leveled) fully homomorphic encryption without bootstrapping. In: *Goldwasser, S. (ed.) ITCS 2012*, pp. 309–325. ACM (2012).
- [9] Zvika Brakerski and Vinod Vaikuntanathan. 2011. Efficient fully homomorphic encryption from (standard) LWE. In: *Ostrovsky, R. (ed.) FOCS 2011*, pp. 97–106. IEEE Computer Society (2011).
- [10] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. 2019. Zether: Towards privacy in a smart contract world. *Cryptology ePrint Archive, Report 2019/191* (2019).
- [11] Zhengjun Cao and LihuaLiu. 2017. The Paillier’s cryptosystem and some variants revisited. *I. J. Network Security* vol. 19(1), pp. 91–98 (2017).
- [12] Dario Catalano, Rosario Gennaro, Nick Howgrave-Graham, and Phong Q. Nguyen. 2001. Paillier’s cryptosystem revisited. In: *Reiter, M.K., Samarati, P. (eds.) CCS 2001*, pp. 206–214. ACM (2001).
- [13] Yu Chen, Xuecheng Ma, Cong Tang, and Man Ho Au. 2019. PGC: Pretty good confidential transaction system with accountability. *Cryptology ePrint Archive, Report 2019/319* (2019). <https://eprint.iacr.org/2019/319>
- [14] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2016. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: *Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I*, LNCS, vol. 10031, pp. 3–33 (2016).
- [15] Don Coppersmith. 1996. Finding a small root of a univariate modular equation. In: *Maurer, U.M. (ed.) EUROCRYPT 1996*, LNCS, vol. 1070, pp. 155–165. Springer (1996).
- [16] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. 1997. A secure and optimally efficient multi-authority election scheme. In: *Fumy, W. (ed.) EUROCRYPT 1997*, LNCS, vol. 1233, pp. 103–118. Springer (1997).
- [17] Ivan Damgård and Mads Jurik. 2001. A generalisation, a simplification and some applications of Paillier’s probabilistic public-key system. In: *Kim, K. (ed.) PKC 2001*, LNCS, vol. 1992, pp. 119–136. Springer (2001).
- [18] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. 2010. A generalization of paillier’s public-key system with applications to electronic voting. *Int. J. Inf. Sec.* vol. 9(6), pp. 371–385 (2010).

Table 3: Performance evaluation of the (basic) Paillier scheme and its additively homomorphic variants. Here we implemented the schemes with $n = 2048$ and $\ell = 448$. “enc. time per msg.” denotes the running time for encrypting a single message, and “dec. time per ct.” the running time for decrypting a single ciphertext.

			The basic Paillier (Scheme 1 in [33])	Jurik's optimization JurOpt[31]		Paillier's optimization PaiOpt(Scheme 3 in [33])		Our optimization NewOpt	
Key Generation AlgorithmKGen		running time (ms)	99.4	97.6		158.6		144.2	
Encryption Algorithm Enc	without any further speed-up	enc. time per msg. (ms)	11.006	5.406		14.124		2.533	
		size of pre-com. table	(do not support pre-com. speed-up)	16 MB	100 MB	16 MB	100 MB	16 MB	100 MB
	time for generating pre-com. table (s)	0.424		2.584	0.427	2.727	0.42	2.716	
	enc. time per msg. (ms)	1.477		1.145	4.751	3.166	0.538	0.422	
Decryption Algorithm	without any further speed-up	dec. time per ct. (ms)	11.05	10.622		2.647		2.520	
Dec	speed-up using CRT	dec. time per ct. (ms)	5.664	5.434		1.384		1.319	

- [19] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. 2012. Multiparty computation from somewhat homomorphic encryption. In: *Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012, LNCS, vol. 7417, pp. 643–662. Springer (2012).*
- [20] C. Ding, D. Pei, and A. Salomaa. 1996. Chinese remainder theorem: Applications in computing, coding, cryptography. *World Scientific Publishing Co., Inc. (1996).*
- [21] Leo Ducas and Daniele Micciancio. 2015. FHEW: bootstrapping homomorphic encryption in less than a second. In: *Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I, LNCS, vol. 9056, pp. 617–640. Springer (2015).*
- [22] Taher El Gamal. 1984. A public key cryptosystem and a signature scheme based on discrete logarithms. In: *Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984, LNCS, vol. 196, pp. 10–18 (1984).*
- [23] Dario Fiore, Rosario Gennaro, and Valerio Pastro. 2014. Efficiently verifiable computation on encrypted data. In: *Ahn, G., Yung, M., Li, N. (eds.) CCS 2014, pp. 844–855. ACM (2014).*
- [24] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. 2000. Sharing decryption in the context of voting or lotteries. In: *Frankel, Y. (ed.) FC 2000, LNCS, vol. 1962, pp. 90–104. Springer (2000).*
- [25] Pierre-Alain Fouque and Jacques Stern. 2001. One round threshold discrete-log key generation without private channels. In: *Kim, K. (ed.) PKC 2001, LNCS, vol. 1992, pp. 300–316. Springer (2001).*
- [26] Rosario Gennaro, Craig Gentry, and Bryan Parno. 2010. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: *Rabin, T. (ed.) CRYPTO 2010, LNCS, vol. 6223, pp. 465–482. Springer (2010).*
- [27] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. 2016. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In: *Manulis, M., Sadeghi, A., Schneider, S. (eds.) ACNS 2016, LNCS, vol. 9696, pp. 156–174. Springer (2016).*
- [28] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In: *Mitzenmacher, M. (ed.) STOC 2009, pp. 169–178. ACM (2009).*
- [29] Shafi Goldwasser and Silvio Micali. 1984. Probabilistic encryption. *Journal of Computer and System Sciences* vol. 28(2), pp. 270–299 (1984).
- [30] Christine Jost, Ha Lam, Alexander Maximov, and Ben Smeets. 2015. Encryption performance improvements of the Paillier cryptosystem. *Cryptology ePrint Archive, Report 2015/864 (2015).* <https://eprint.iacr.org/2015/864>
- [31] Mads Jurik. 2003. Extensions to the Paillier cryptosystem with applications to cryptographic protocols. *PhD Thesis (2003).*
- [32] Yehuda Lindell. 2017. Fast secure two-party ECDSA signing. In: *Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II, LNCS, vol. 10402, pp. 613–644. Springer (2017).*
- [33] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In: *Stern, J. (ed.) EUROCRYPT 1999, LNCS, vol. 1592, pp. 223–238 (1999).*
- [34] J.M. Pollard. 1974. Theorems on factorization and primality testing. In: *Mathematical Proceedings of the Cambridge Philosophical Society, vol. 76, pp. 521–528. Cambridge University Press (1974).*
- [35] Kouichi Sakurai and Tsuyoshi Takagi. 2002. New semantically secure public-key cryptosystems from the RSA-primitive. In: *Naccache, D., Paillier, P. (eds.) PKC 2002, LNCS, vol. 2274, pp. 1–16. Springer (2002).*
- [36] Hossein Shafagh, Anwar Hithnawi, Andreas Driescher, Simon Duquenois, and Wen Hu. 2015. Talos: Encrypted query processing for the internet of things. In: *Song, J., Abdelzaher, T.F., Mascolo, C. (eds.) Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, SenSys 2015, Seoul, South Korea, November 1–4, 2015, pp. 197–210. ACM (2015).*
- [37] Victor Shoup. 2006. A computational introduction to number theory and algebra. *Cambridge University Press (2006).*

A OMITTED PROOFS

We present the proofs for the correctness and the additive homomorphism of our optimization NewOpt.

THEOREM 6 (CORRECTNESS). *For all possible messages $m \in \mathbb{Z}_N$ and randomnesses $r \in \{0, 1\}^\ell$, the decryption (4) always undoes the encryption (3).*

PROOF. Let $c = (1 + N)^m \cdot (h^r \bmod N)^N \bmod N^2$ be the encryption of m under randomness r . By Lemma 1, we could write c using the isomorphism ψ , i.e., $c = \psi(m, h^r)$. Thus, we have

$$\begin{aligned} c^{2\alpha} \bmod N^2 &= \psi(m, h^r)^{2\alpha} \\ &= \psi(2\alpha m, h^{2\alpha r}) \end{aligned} \quad (21)$$

$$= \psi(2\alpha m, y^{4\alpha\beta r}) \quad (22)$$

$$= \psi(2\alpha m, y^{(P-1)(Q-1)r}) \quad (23)$$

$$= \psi(2\alpha m, 1) \quad (24)$$

$$= (1 + N)^{2\alpha m},$$

where (21) holds since ψ is an isomorphism between $\mathbb{Z}_N \times \mathbb{Z}_N^*$ and $\mathbb{Z}_{N^2}^*$, (22) follows from the fact that $h = -y^{2\beta} \bmod N$, (23) from $4\alpha\beta = (P-1)(Q-1)$, and (24) from $(P-1)(Q-1) = |\mathbb{Z}_N^*|$ which annihilates $y \in \mathbb{Z}_N^*$.

Consequently, in the decryption,

$$L(c^{2\alpha} \bmod N^2, N) \cdot (2\alpha)^{-1} \bmod N = 2\alpha m \cdot (2\alpha)^{-1} = m, \quad (25)$$

where the first equality in (25) is guaranteed by the functionality of the algorithm L (cf. Lemma 1). This completes the proof of Theorem 6. $\square$

THEOREM 7 (ADDITIVE HOMOMORPHISM). *For all possible messages $m_1, m_2 \in \mathbb{Z}_N$ and randomnesses $r_1, r_2 \in \{0, 1\}^\ell$, it holds that $\text{Enc}_{\text{NewOpt}}(pk, m_1; r_1) \cdot \text{Enc}_{\text{NewOpt}}(pk, m_2; r_2) = \text{Enc}_{\text{NewOpt}}(pk, m_1 + m_2; r_1 + r_2)$.*

PROOF. It is easy to check the above equality by using the isomorphism ψ defined in Lemma 1. More precisely,

$$\begin{aligned}
 & \text{Enc}_{\text{NewOpt}}(pk, m_1; r_1) \cdot \text{Enc}_{\text{NewOpt}}(pk, m_2; r_2) \\
 = & (1 + N)^{m_1} \cdot (h^{r_1} \bmod N)^N \cdot (1 + N)^{m_2} \cdot (h^{r_2} \bmod N)^N \pmod{N^2} \\
 = & \psi(m_1, h^{r_1}) \cdot \psi(m_2, h^{r_2}) \\
 = & \psi(m_1 + m_2, h^{r_1+r_2}) \\
 = & (1 + N)^{m_1+m_2} \cdot (h^{r_1+r_2} \bmod N)^N \pmod{N^2} \\
 = & \text{Enc}_{\text{NewOpt}}(pk, m_1 + m_2; r_1 + r_2),
 \end{aligned} \tag{26}$$

where (26) follows from the isomorphism of ψ . This completes the proof. $\square$ $\square$

Figure 3: The performance of the KGen algorithm.

B OMITTED FIGURES

Figure 4: Left figure: The performance of the Enc algorithm. Here “# of enc. per second” denotes the number of encryptions could be made within a second. We give three flavors of implementations. Left: the straightforward implementation without any further speed-up; Middle: the implementation by pre-computation, using a 16 MB pre-computed table; Right: the implementation by pre-computation, using a 100 MB pre-computed table. We note that the speed-up technique by pre-computation is not applicable to the basic Paillier scheme. **Right figure: The performance of the Dec algorithm.** Here “# of dec. per second” denotes the number of decryptions could be made within a second. We give two flavors of implementations. Left: the straightforward implementation without any further speed-up; Right: the implementation by applying the Chinese Remainder Theorem (CRT).