How to Remove the Exponent GCD in HK09*

Xianhui Lu, Bao Li, and Yamin Liu

Institute of Information Engineering of Chinese Academy of Sciences, Beijing, 100093, China
{xhlu,ymliu,lb}@is.ac.cn

Abstract. To improve the decapsulation efficiency of HK09 (proposed by Hofheinz and Kiltz in Eurocrypt 2009), we propose a new skill to remove the exponent GCD operation. In the proposed scheme, the decapsulation efficiency is improved by 38.9% (instantiated over the semi-smooth subgroup) and the efficiency of encapsulation is dropped by 5.7%.

Keywords: public key encryption, chosen ciphertext security, factoring.

1 Introduction

Based on the Blum-Goldwasser encryption (BG84) [2], Hofheinz and Kiltz proposed the first practical IND-CCA (Chosen Ciphertext Attack) secure public key encryption scheme from the factoring assumption [7](HK09) in the standard model. The BG84 scheme is IND-CPA (Chosen Plaintext Attack) secure under the factoring assumption. To achieve IND-CCA security, Hofheinz and Kiltz used the famous All-But-One skill [6,3,4,8], which was widely used in the construction of IND-CCA secure encryption schemes.

The skill of HK09 was later generalized to the extractable hash proof system by Wee in [13]. In [13], Wee also proposed a conceptually simpler variant of HK09 which is more modular but less efficient (there is a linear blow-up in both ciphertext overhead and public key size over HK09).

The efficiency of HK09 was later improved by Mei [11] and Lu [9,10]. In [11], the authors instantiated HK09 over the semi-smooth subgroup and also proposed an ElGamal style variant of HK09. Briefly, semi-smooth subgroup consider the modulus of $N = PQ = (2p'p+1)(2q'q+1)$, where (p', q') are prime numbers large enough but much smaller than (P, Q), and (p, q) are product of distinct prime numbers smaller than a bound. The unique subgroup of QR_N (the quadratic residuosity group) with order $p'q'$ is called semi-smooth subgroup. Since $p'q'$ is much smaller than the order of QR_N, schemes instantiated over semi-smooth subgroup are more efficient. In [9] the authors proposed a tradeoff between the

* Supported by the National Basic Research Program of China (973 project)(No.2013CB338002), the National Nature Science Foundation of China (No.61070171, No.61272534), the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA06010702, IIE's Cryptography Research Project (No.Y3Z0024103, Y3Z0027103).

W. Susilo and R. Reyhanitabar (Eds.): ProvSec 2013, LNCS 8209, pp. 239–248, 2013.

efficiency of encapsulation and decapsulation of HK09. The efficiency of decapsulation was improved by 38.9% and the efficiency of encapsulation was dropped by 11.4% (instantiated over the semi-smooth subgroup). In [10] the authors improved the decapsulation efficiency at the price of a slightly increased key size. The decapsulation efficiency is improved by 32% (instantiated over the quadratic residuosity group) or 57.6% (instantiated over the semi-smooth subgroup) and the encapsulation efficiency remains the same.

1.1 Motivation

The ciphertext of HK09 is $(R = g^{\mu 2^{l_K}+l_H}, S = |g^{\mu t} X^{\mu}|)$, the encapsulated key is $K = \mathrm{BBS}_r(g^{\mu 2^{l_H}})$, where l_K is the length of K, l_H is the length of the hash value $t = \mathrm{H}(R)$, $\mathrm{BBS}_r()$ is a Blum-Blum-Shub pseudorandom generator [1]. Since the exponent inversion can not be computed directly for hidden order group, the decapsulation algorithm computes $g^{\mu 2^{l_H}}$ by using Shamir's GCD (greatest common divisor) in the exponent algorithm [12].

One of the skills to improve the efficiency of HK09 is to remove the exponent GCD operation in the decapsulation. In [9] the authors derive the encapsulated key from $g^{\mu t 2^{l_H}}$ and compute $K = \mathrm{BBS}_r((S/R^{\rho})^{2^{l_H}})$ directly. In [10] the authors remove the computation of exponent GCD by hiding g^{μ} instead of $g^{\mu t}$ into S.

The above skills to remove the exponent GCD operation also have some drawbacks. The skill used in [9] causes a loose security reduction and the skill used in [10] increases the size of the key.

An interesting question is, how can we remove the exponent GCD operation while maintain the key size and the security reduction complexity?

1.2 Our Contribution

We propose a new method to remove the exponent GCD operation in HK09. The decapsulation efficiency is improved by 38.9% (instantiated over the semi-smooth subgroup) and the efficiency of encapsulation is dropped by 5.7%.

Our main idea is to directly embed g^{μ} into S. Concretely, the ciphertext is $(R = g^{\mu 2^{l_K}}, S = |g^{\mu} X^{\mu t}|)$, the encapsulated key is $K = \mathrm{BBS}_N(g^{\mu})$, where $g \in \mathrm{QR}_N, X = g^{x 2^{l_K}}$, $x \in [(N-1)/4]$ is the private key. Thus, the decapsulation computes $g^{\mu} = S/R^{xt}$ directly.

One of the main difficulties in the security proof is the construction of the challenge ciphertext. According the All-But-One skill, the simulator needs to set $X = g^{x 2^{l_K}} g^{-1/t^*}$. Unfortunately, the simulator can not compute $1/t^*$ since he does not know the factoring of N. Our solution is to choose $h \in QR_N$ and set $g = h^{t^*}$. Thus the simulator can set $X = g^{x 2^{l_K}} h^{-1}$.

The other difficulty in the security reduction is the simulation of the decapsulation operation. When the adversary submits a ciphertext $(R = g^{\mu 2^{l_K}}, S = |g^{\mu} X^{\mu t}|)$, the simulator can compute $(S/R^{xt})^{t^*} = g^{\mu(t^*-t)}$ and then get $g^{\mu 2^c}$, where $2^c = gcd(2^{l_K}, (t^* - t))$. If $c \geq 1$, the simulator can not compute g^{μ}. To

solve this problem we use the same skill as in [7]. Briefly, the simulator sets $R = g^{\mu 2^{l_K + l_H}}$ and computes $K = \mathrm{BBS}_N(g^{\mu 2^{l_H}})$.

Compared with the scheme in [9], the encapsulation of our new scheme is more efficient and the efficiency of decapsulation remains the same. More importantly, the security reduction of our new scheme is tighter. Compared with the scheme in [10], their scheme is more efficient, while the key of our new scheme is shorter.

We remark that our new variant can be instantiated over the semi-smooth subgroup using the technique in [11]. The resulting scheme is more efficient than that over the QR_N group.

1.3 Outline

In section 2 we review the definition of key encapsulation mechanism and target collision resistant hash function. In section 3 we propose our new variant of HK09. Finally we give the conclusion in section 4.

2 Definitions

In describing probabilistic processes, $x \xleftarrow{R} X$ denotes that x is sampled according to the distribution X. If S is a finite set, $s \xleftarrow{R} S$ denotes that s is sampled from the uniform distribution on S. If A is a probabilistic algorithm and x an input, then $A(x)$ denotes the output distribution of A on input x. Thus, we write $y \xleftarrow{R} A(x)$ to denote of running algorithm A on input x and assigning the output to the variable y.

2.1 Key Encapsulation Mechanism

A key encapsulation mechanism consists of the following algorithms:

- KEM.KeyGen(1^k): A probabilistic polynomial-time key generation algorithm takes as input a security parameter (1^k) and outputs a public key PK and a secret key SK. We write $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathrm{KEM.KeyGen}(1^k)$
- KEM.Enc(PK): A probabilistic polynomial-time encapsulation algorithm takes as input the public key PK, and outputs a pair (K, ψ), where $K \in K_D$(K_D is the key space) is a key and ψ is a ciphertext. We write $(K, \psi) \leftarrow \mathrm{KEM.Enc(PK)}$
- KEM.Dec(SK, ψ): A decapsulation algorithm takes as input a ciphertext ψ and the secret key SK. It returns a key K. We write $K \leftarrow \mathrm{KEM.Dec}(\mathrm{SK}, \psi)$.

We require that for all (PK,SK) output by KEM.KeyGen(1^k), all $(K, \psi) \in$ [KEM.Enc(PK)], we have KEM.Dec(SK, ψ)=K.

Now we review the IND-CCA (Indistinguishability against adaptive chosen ciphertext attack) security of KEM. Note that we use the definition in [8] which is simpler than the original definition in [5].

Definition 1. *A KEM scheme is secure against adaptive chosen ciphertext attacks if the advantage of any adversary in the following game is negligible in the security parameter k.*

1. The adversary queries a key generation oracle. The key generation oracle computes $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathrm{KEM.KeyGen}(1^k)$ and responds with PK.
2. The adversary queries an encapsulation oracle. The encapsulation oracle computes:

$$b \overset{R}{\leftarrow} \{0,1\}, (K_1, \psi^*) \leftarrow \mathrm{KEM.Enc}(\mathrm{PK}), K_0 \overset{R}{\leftarrow} K_D,$$

   and responds with (K_b, ψ^*).
3. The adversary makes a sequence of calls to the decapsulation oracle. For each query the adversary submits a ciphertext ψ, and the decapsulation oracle responds with $\mathrm{KEM.Dec}(\mathrm{SK}, \psi)$. The only restriction is that the adversary can not request the decapsulation of ψ^*.
4. Finally, the adversary outputs a guess b'.

The adversary's advantage in the above game is $\mathrm{Adv}_{\mathrm{A}}^{\mathrm{cca}}(k) = |\Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]|$. If a KEM is secure against adaptive chosen ciphertext attacks defined in the above game we say it is IND-CCA secure.

2.2 Target Collision Resistant Hash Function

Now we review the definition of target collision resistant (TCR) hash function. We say that a function $\mathrm{H} : X \to Y$ is a TCR hash function, if given a random preimage $x \in X$, it is hard to find $x' \neq x$ with $\mathrm{H}(x') = \mathrm{H}(x)$. Concretely, the advantage of an adversary $\mathcal{A}$ is defined as:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{tcr}}(k) = \Pr[x \overset{R}{\leftarrow} X, x' \leftarrow \mathrm{A}(x) : x \neq x' \wedge \mathrm{H}(x) = \mathrm{H}(x')].$$

We say H is a TCR hash function if $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{tcr}}(k)$ is negligible.

3 New Variant of HK09

Our new variant of HK09 is described as follows.

– KeyGen: The key generation algorithm chooses uniformly at random a Blum integer $N = PQ = (2p+1)(2q+1)$, where P, Q, p, q are prime numbers, then computes:

$$g \overset{R}{\leftarrow} \mathrm{QR}_N, x \overset{R}{\leftarrow} [(N-1)/4], X \leftarrow g^{x 2^{l_K + l_H}},$$

$$pk \leftarrow (N, g, X), sk \leftarrow x,$$

where $\mathrm{H} : QR_N \to \{0,1\}^{l_H}$ is a TCR hash function, l_H is the bit length of the output value of H, l_K is the bit length of the encapsulated key K, .

– Encapsulation: Given pk, the encapsulation algorithm computes:

$$\mu \xleftarrow{R} [(N-1)/4], R \leftarrow g^{\mu 2^{l_K+l_H}}, t \leftarrow \mathrm{H}(R), S \leftarrow \left|\left(gX^t\right)^{\mu}\right|,$$

$$K \leftarrow \mathrm{BBS}_N(g^{\mu 2^{l_H}}),$$

where $\mathrm{BBS}_N(\alpha) = \mathrm{LSB}(\alpha), \cdots, \mathrm{LSB}(\alpha^{2^{l_K-1}})$, $\mathrm{LSB}(\alpha)$ denotes the least significant bit of α.

– Decapsulation: Given a ciphertext (R, S) and sk, the decapsulation algorithm verifies $R \in Z_N^*, S \in Z_N^* \cap [(N-1)/2]$, then computes:

$$t \leftarrow \mathrm{H}(R), \rho \leftarrow xt,$$

$$\text{if } \left(\frac{S}{R^{\rho}}\right)^{2^{l_K+l_H}} = R \text{ then computes } K \leftarrow \mathrm{BBS}_N\left(\frac{S^{2^{l_H}}}{R^{\rho 2^{l_H}}}\right),$$

$$\text{else returns the rejection symbol } \perp .$$

The correctness of the scheme above can be verified as follows:

$$\left(\frac{S^{2^{l_H}}}{R^{\rho 2^{l_H}}}\right) = \left(\frac{|(gX^t)^{\mu}|^{2^{l_H}}}{(g^{\mu 2^{l_K+l_H}})^{xt 2^{l_H}}}\right) = \left(\frac{|(g(g^{x2^{l_K+l_H}})^t)^{\mu}|^{2^{l_H}}}{(g^{\mu 2^{l_K+l_H}})^{xt 2^{l_H}}}\right) = g^{\mu 2^{l_H}}.$$

We remark that, similar to [10], if pq is added to the private key, the efficiency of decapsulation can be improved by computing $\rho = xt \mod pq$. It is clear that our new variant above can also be instantiated over semi-smooth subgroup using the technique in [11]. In this case, x is selected from $2^{l_{p'}+l_{q'}+\lambda}$, where $l_{p'}$ is the length of p', $l_{q'}$ is the length of q', λ is a parameter for security level. If $p'q'$ is added to the private key, the efficiency of decapsulation can be further improved by selecting x from $[p'q']$ instead of $2^{l_{p'}+l_{q'}+\lambda}$.

3.1 Security Proof

Theorem 1. *If factoring N is hard and H is a TCR hash function, then the new variant is IND-CCA secure.*

The proof is similar to that of HK09, in which the reduction is divided into two phases. First, the BBS distinguisher is reduced to the factoring assumption. Then, the IND-CCA security of the scheme is reduced to the BBS distinguisher. The experiment for the BBS distinguish problem is defined as:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{BBS}} = |\Pr[\mathcal{A}(N, z, \mathrm{BBS}_N(u)) = 1] - \Pr[\mathrm{A}(N, z, U) = 1]|,$$

where N is a Blum integer ($N = PQ, P = 2p+1, Q = 2q+1$, p and q are prime numbers), $u \in QR_N$, $z = u^{2^{l_K}}$, U is a random bit string of length l_K.

Given Theorem 2 in [7], it is clear that we only need to prove the following theorem.

Theorem 2. *If it is hard to distinguish* $(N, z, \mathrm{BBS}_N(u))$ *from* (N, z, U) *and H is a TCR hash function, then the new variant is IND-CCA secure.*

Proof. Suppose that an adversary $\mathcal{A}$ can break the IND-CCA security of the new variant. To prove the theorem, we construct an adversary $\mathcal{B}$ to distinguish $(N, z, \mathrm{BBS}_N(u))$ from (N, z, U). The construction of $\mathcal{B}$ is described as follows.

Setup: On receiving (N, z, V), where $V = U$ or $V = \mathrm{BBS}_N(u)$, the adversary $\mathcal{B}$ computes:

$$t^* \leftarrow \mathrm{H}(z), h \xleftarrow{R} \mathrm{QR}_N, g \leftarrow h^{t^*}, x \xleftarrow{R} [(N-1)/4],$$

$$X \leftarrow g^{x2^{l_K+l_H}} h^{-1}, pk \leftarrow (N, g, X).$$

The adversary $\mathcal{B}$ sends pk to adversary $\mathcal{A}$.

Challenge: The adversary $\mathcal{B}$ constructs the challenge ciphertext as follows.

$$R^* \leftarrow z, S^* \leftarrow \left| R^{*xt^*} \right|, K^* \leftarrow V.$$

Let $R^* = g^{\mu^* 2^{l_K+l_H}}$, the correctness of the challenge ciphertext can be verified as follow:

$$\begin{aligned} S^* &= \left| R^{*xt^*} \right| \\ &= \left| g^{\mu^* 2^{l_K+l_H}(xt^*)} \right| \\ &= \left| g^{\mu^*} g^{\mu^* 2^{l_K+l_H} xt^*} g^{-\mu^*} \right| \\ &= \left| g^{\mu^*} (g^{x2^{l_K+l_H}} h^{-1})^{\mu^* t^*} \right| \\ &= \left| g^{\mu^*} X^{\mu^* t^*} \right| \\ &= \left| (gX^{t^*})^{\mu^*} \right|. \end{aligned} \tag{1}$$

Decapsulation: On receiving the decapsulation query (R, S), the adversary $\mathcal{B}$ verifies $R \in Z_N^*, S \in Z_N^* \cap [(N-1)/2]$, then computes:

$$t \leftarrow \mathrm{H}(R).$$

Then the adversary $\mathcal{B}$ considers three cases:

Case 1: $t \neq t^*$. In this case, the adversary $\mathcal{B}$ acts as:

$$\text{if } \left(\frac{S}{R^{xt}}\right)^{t^* 2^{l_K+l_H}} = R^{(t^*-t)} \text{ computes:}$$

$$2^c = \gcd(t^* - t, 2^{l_K+l_H}) = a(t^* - t) + b2^{l_K+l_H},$$

$$\text{returns } K \leftarrow \mathrm{BBS}_N \left(\left(\left(SR^{-xt} \right)^{t^* a} R^b \right)^{2^{l_H-c}} \right),$$

$$\text{else returns the rejection symbol } \perp .$$

Since $t \neq t^*$ we have $0 < c < l_H$. Let $R = g^{\mu 2^{l_K + l_H}}$, the correctness of the verification equation can be verified as follows:

$$\begin{aligned}
\left(\frac{S}{R^{xt}}\right)^{t^* 2^{l_K+l_H}} &= \left(\frac{(gX^t)^\mu}{g^{\mu x t 2^{l_K+l_H}}}\right)^{t^* 2^{l_K+l_H}} \\
&= \left(\frac{(g g^{xt 2^{l_K+l_H}} h^{-t})^\mu}{g^{\mu x t 2^{l_K+l_H}}}\right)^{t^* 2^{l_K+l_H}} \\
&= \left((gh^{-t})^\mu\right)^{t^* 2^{l_K+l_H}} \\
&= \left(g^{t^*} g^{-t}\right)^{\mu 2^{l_K+l_H}} \\
&= g^{(t^*-t)\mu 2^{l_K+l_H}} \\
&= R^{(t^*-t)}.
\end{aligned} \tag{2}$$

The correctness of K can be verified as follows:

$$\begin{aligned}
K &= \mathrm{BBS}_N\left(\left((SR^{-xt})^{t^* a} R^b\right)^{2^{l_H - c}}\right) \\
&= \mathrm{BBS}_N\left(\left(\left(\frac{(gX^t)^\mu}{g^{\mu x t 2^{l_K+l_H}}}\right)^{t^* a} R^b\right)^{2^{l_H - c}}\right) \\
&= \mathrm{BBS}_N\left(\left(\left(\frac{(g g^{xt 2^{l_K+l_H}} h^{-t})^\mu}{g^{\mu x t 2^{l_K+l_H}}}\right)^{t^* a} R^b\right)^{2^{l_H - c}}\right) \\
&= \mathrm{BBS}_N\left(\left((gh^{-t})^\mu\right)^{t^* a} R^b\right)^{2^{l_H - c}}\right) \\
&= \mathrm{BBS}_N\left(\left(g^{\mu(t^*-t)a} g^{\mu 2^{l_K+l_H} b}\right)^{2^{l_H - c}}\right) \\
&= \mathrm{BBS}_N\left(\left(g^{\mu(a(t^*-t)+b 2^{l_K+l_H})}\right)^{2^{l_H - c}}\right) \\
&= \mathrm{BBS}_N\left(\left(g^{\mu 2^c}\right)^{2^{l_H - c}}\right) \\
&= \mathrm{BBS}_N\left(g^{\mu 2^{l_H}}\right).
\end{aligned} \tag{3}$$

Case 2: $t = t^*, R \neq R^*$. Denote this case as an event $\mathrm{bad}_{\mathrm{tcr}}$. Since H is a TCR hash function, we have $\Pr[\mathrm{bad}_{\mathrm{tcr}}] \leq \mathrm{Adv}^{\mathrm{tcr}}_{\mathcal{C}}$.

Case 3: $t = t^*, R = R^*, S \neq S^*$. In this case, if $S^2 \neq R^{2xt}$ return the rejection symbol $\perp$. If $S^2 = R^{2xt}$, we have $|S| = S \neq S^* = |S^*|$ and $S^2 = R^{2xt} = R^{*2xt^*} = S^{*2}$. Then, $S \neq \pm S^*$ and $S^2 - S^{*2} = (S + S^*)(S - S^*) = 0$. Thus $\mathcal{B}$ can factor N directly by computing $\gcd(N, S + S^*)$ or $\gcd(N, S - S^*)$.

Guess: On receiving b' from adversary $\mathcal{A}$, the adversary $\mathcal{B}$ outputs b'.

This finishes the construction of the adversary $\mathcal{B}$. We claim that the distribution of simulated public key and the challenge ciphertext are almost identical in the simulation above and the IND-CCA game.

Lemma 1. *There exists an event* $\mathrm{bad}_{\mathrm{key}}$ *such that, conditioned on* $\neg\mathrm{bad}_{\mathrm{key}}$ *the public key and the challenge ciphertext are identically distributed in simulation and the IND-CCA game. Concretely,*

$$\Pr[\text{bad}_{\text{key}}] \leq \frac{5}{2^{k-1}},$$

where k is the parameter of security level.

Since the proof of the lemma above is very similar to that of lemma 1 in [7], we omit the detail.

It is clear that, unless bad_{tcr} or bad_{key} occurs, $\mathcal{B}$ perfectly simulates the real IND-CCA game. To be concrete:

$$\begin{aligned} \text{Adv}_{\mathcal{B}}^{\text{BBS}} &= \text{Adv}_{\mathcal{A}}^{\text{cca}} - \Pr[\text{bad}_{\text{tcr}}] - \Pr[\text{bad}_{\text{key}}] \\ &\geq \text{Adv}_{\mathcal{A}}^{\text{cca}} - \text{Adv}_{\mathcal{C}}^{\text{tcr}} - \tfrac{5}{2^{k-1}}. \end{aligned} \tag{4}$$

This completes the proof of theorem 2. □

3.2 Efficiency

In this section, we analyze the efficiency of our new variant and compare it with the previous schemes in [7,11,9,10]. Note that, all of these schemes can be instantiated over the QR_N group or the semi-smooth subgroup. For the sake of clarity, these two cases are discussed respectively.

The Case of QR_N Group. The efficiency of schemes in [7,11,9,10] and our variant is listed in table 1, where HK09 is the scheme in [7], E-HK is the ElGamal style variant of HK09 in [11], LLML2011 is the variant of HK09 in [9], LLML2012 is the variant of HK09 in [10] and NEW is the proposed variant. The parameters are the same as those in [7,11,9,10], $l_N = 1024, l_K = l_H = 80$.

Table 1. Schemes instantiated over the QR_N group

	Encapsulate(mul)	Decapsulate(mul)	SK (bits)	PK (bits)
HK09	$3272(3l_N + l_K + 1.5l_H)$	$2376(1.5l_N + 4l_K + 6.5l_H)$	l_N	$2l_N$
E-HK	$4808(4.5l_N + l_K + 1.5l_H)$	$2043(1.5 \times 1.2l_N + 2.5l_H)$	$2l_N$	$3l_N$
LLML2011	$3432(3l_N + 2l_K + 2.5l_H)$	$1816(1.5l_N + l_K + 2.5l_H)$	l_N	$2l_N$
LLML2012	$3272(3l_N + l_K + 1.5l_H)$	$1736(1.5l_N + l_K + 1.5l_H)$	$2l_N$	$3l_N$
NEW	$3352(3l_N + l_K + 2.5l_H)$	$1816(1.5l_N + l_K + 2.5l_H)$	l_N	$2l_N$

The encapsulation of our variant can first compute $A = g^{\mu}$, which requires $1.5l_N$ multiplications. Then, the computation of $B = X^{\mu t}$ requires $1.5l_N + 1.5l_H$ multiplications. Finally, the computations of $R = A^{2^{l_K + l_H}} = g^{2^{l_K + l_H}\mu}$ and $K = \text{BBS}_N(A^{2^{l_K}})$ require $l_K + l_H$ multiplications. Thus, the encapsulation requires $3l_N + l_K + 2.5l_H$ multiplications. The decapsulation computes $D = R^{\rho}$, which requires $1.5l_N + 1.5l_H$ multiplications (the length of $\rho = xt$ is $l_N + l_H$). Then computes $(S/D)^{2^{l_K + l_H}}$ and $K = \text{BBS}_N((S/D)^{2^{l_H}})$, which require $l_K + l_H$ multiplications. We have that the decapsulation requires $1.5l_N + l_K + 2.5l_H$

multiplications. Note that, the decapsulation can be improved by adding pq to the private key and computing $\rho = xt \mod pq$. As a result, the decapsulation requires $1.5l_N + l_K + l_H$ multiplications.

The Case of Semi-smooth Subgroup Group. The efficiency of schemes in [7,11,9,10] and our variant is listed in table 2, where S-HK is the instantiation of HK09, S-E-HK is the instantiation of E-HK, S-LLML2011 is the instantiation of LLML2011, S-LLML2012 is the instantiation of LLML2012 and S-NEW is the instantiation of our new variant. The parameters are the same as those in [7,11,9], $l_K = l_H = 80, l_{p'} = l_{q'} = 160, \lambda = 80, l_e = l_{p'} + l_{q'} + \lambda = 400, l_{e'} = l_{p'} + l_{q'} = 320$.

Table 2. Schemes instantiated over the semi-smooth subgroup

	Encapsulate(mul)	Decapsulate(mul)	SK (bits)	PK (bits)
S-HK	$1400(3l_e + l_K + 1.5l_H)$	$1440(1.5l_e + 4l_K + 6.5l_H)$	l_e	$2l_N$
S-E-HK	$2000(4.5l_e + l_K + 1.5l_H)$	$920(1.5 \times 1.2l_e + 2.5l_H)$	$2l_e$	$3l_N$
S-LLML2011	$1560(3l_e + 2l_K + 2.5l_H)$	$880(1.5l_e + l_K + 2.5l_H)$	l_e	$2l_N$
S-LLML2012	$1400(3l_e + l_K + 1.5l_H)$	$800(1.5l_e + l_K + 1.5l_H)$	$2l_e$	$3l_N$
S-NEW	$1480(3l_e + l_K + 2.5l_H)$	$880(1.5l_e + l_K + 2.5l_H)$	l_e	$2l_N$

Note that, the private key of schemes instantiated over semi-smooth subgroup is selected from $[2^{l_{p'}+l_{q'}+\lambda}]$. When $p'q'$ is added to the private key, the decapsulation efficiency can be improved by selecting the private key from $[p'q']$.

4 Conclusion

We proposed a new method to remove the exponent GCD operation in HK09, which improves the decapsulation without increasing the key size. The decapsulation efficiency is improved by 38.9% (instantiated over the semi-smooth subgroup) and the efficiency of encapsulation is dropped by 5.7%. Compared with previous skill in [9] to remove the exponent GCD operation, the security reduction of our new scheme is tighter. Compared with the skill in [10], their scheme is more efficient, while the key of our new scheme is shorter. We proved that the proposed variant is IND-CCA secure under the factoring assumption.

References

1. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo-random number generator. SIAM J. Comput. 15(2), 364–383 (1986)
2. Blum, M., Goldwasser, S.: An probabilistic public key encryption scheme which hides all partial information. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 289–299. Springer, Heidelberg (1985)
3. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

4. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: ACM Conference on Computer and Communications Security, pp. 320–329. ACM (2005)
5. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput. 33, 167–226 (2004), http://dl.acm.org/citation.cfm?id=953065.964243
6. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: STOC, pp. 542–552. ACM (1991)
7. Hofheinz, D., Kiltz, E.: Practical chosen ciphertext secure encryption from factoring. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 313–332. Springer, Heidelberg (2009)
8. Kiltz, E.: Chosen-ciphertext secure key-encapsulation based on gap hashed diffie-hellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (2007)
9. Lu, X., Li, B., Mei, Q., Liu, Y.: Improved tradeoff between encapsulation and decapsulation of HK09. In: Wu, C.-K., Yung, M., Lin, D. (eds.) Inscrypt 2011. LNCS, vol. 7537, pp. 131–141. Springer, Heidelberg (2012)
10. Lu, X., Li, B., Mei, Q., Liu, Y.: Improved efficiency of chosen ciphertext secure encryption from factoring. In: Ryan, M.D., Smyth, B., Wang, G. (eds.) ISPEC 2012. LNCS, vol. 7232, pp. 34–45. Springer, Heidelberg (2012)
11. Mei, Q., Li, B., Lu, X., Jia, D.: Chosen ciphertext secure encryption under factoring assumption revisited. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 210–227. Springer, Heidelberg (2011)
12. Shamir, A.: On the generation of cryptographically strong pseudo-random sequences. In: Even, S., Kariv, O. (eds.) ICALP 1981. LNCS, vol. 115, pp. 544–550. Springer, Heidelberg (1981)
13. Wee, H.: Efficient chosen-ciphertext security via extractable hash proofs. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 314–332. Springer, Heidelberg (2010)