Practical Chosen Ciphertext Secure Encryption from Factoring

Dennis Hofheinz[⋆] and Eike Kiltz[⋆⋆]

Cryptology & Information Security Group
CWI Amsterdam, The Netherlands
{hofheinz,kiltz}@cwi.nl

Abstract. We propose a practical public-key encryption scheme whose security against chosen-ciphertext attacks can be reduced in the standard model to the assumption that factoring is intractable.

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

1 Introduction

The security of almost any cryptographic primitive (such as public-key encryption or digital signatures) has to rely on the computational hardness of a certain number-theoretic problem. Unfortunately, since there are currently no tools available to rigorously prove lower bounds on the complexity of such problems, one has to base security on (unproven) *cryptographic hardness assumptions.* The only confidence we have in such assumptions is that after a sufficiently large period of time, nobody could successfully refute them. The most established cryptographic hardness assumption is without doubt the so called *factoring assumption* which states that, given the product of two distinct large primes, it is computationally infeasible to reconstruct the primes. Despite of intensive research, no algorithm has been found that can efficiently factor composite numbers.

Main result. In this paper we propose a new public-key encryption scheme that is based on Rabin's trapdoor one-way permutation [35]. We can prove that the security of our scheme against adaptive chosen-ciphertext attacks (CCA security) is equivalent to the factoring assumption. Furthermore, the scheme is practical as its encryption performs only roughly two, and its decryption roughly one modular exponentiation. To the best of our knowledge, this is the first scheme that simultaneously enjoys those two properties.

History. The notion of CCA security is due to Rackoff and Simon [36] and is now widely accepted as the standard security notion for public-key encryption schemes. In contrast to security against passive adversaries (security against

[⋆] Supported by the Dutch Organization for Scientific Research (NWO).

[⋆⋆] Supported by the research program Sentinels (http://www.sentinels.nl). Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

chosen-plaintext attacks aka semantic security), in a chosen-ciphertext attack the adversary plays an active role by obtaining the decryptions of ciphertexts (or even arbitrary bit-strings) of his choosing. The practical significance of such attacks was demonstrated by Bleichenbacher [4] by means of a CCA attack against schemes following the encryption standard PKCS #1.

Historically, the first scheme that was provably secure against CCA attacks is due to Dolev, Dwork, and Naor [17] (building on an earlier result by Naor and Yung [31]). Their generic construction is based on enhanced trapdoor permutations and therefore (using the enhanced trapdoor permutations from [20, App. C]) yields a scheme CCA secure under the factoring assumption. However, in practice these schemes are prohibitively impractical, as they rely on expensive non-interactive zero-knowledge proofs. The first practical schemes provably CCA secure under standard cryptographic hardness assumptions were due to Cramer and Shoup [15,14]. However, their framework of “hash proof systems” inherently relies on *decisional assumptions* such as the assumed hardness of deciding if a given integer has a square root modulo a composite number with unknown factorization (DQR assumption), or of deciding if a given tuple is a Diffie-Hellman tuple or not (DDH assumption). Until today, Cramer and Shoup’s framework of hash proof systems (with its variations from [29,19,11,28,24,27]) and the recent concept of lossy trapdoor functions [33] yield the only known CCA secure practical encryption schemes based on an assumption related to factoring: the DQR assumption and Paillier’s decisional composite residuosity (DCR) assumption. Currently, no practical scheme is known that is CCA secure solely under the factoring assumption (or even under the potentially stronger RSA assumption).

In general, decisional assumptions are a much stronger class of assumptions than computational assumptions. For example, deciding if a given integer has a modular square root or not may be much easier than actually computing a square root (or, equivalently, factoring the modulus). It is noteworthy that there are known ways to achieve CCA security that do not inherently rely on decisional assumptions (e.g., [9,13,23]). In particular, the first practical encryption scheme CCA secure under the *Computational* Diffie-Hellman (CDH) assumption was only recently proposed by Cash, Kiltz, and Shoup [13] and improved by Hanaoka and Kurosawa [23]. On the other hand, [9] provide a practical encryption scheme CCA secure under the Bilinear Computational Diffie-Hellman (BCDH) assumption.

Random oracle schemes. In a different line of research, Bellare and Rogaway [2,3] presented practical schemes for which they give heuristic proofs of CCA security under standard computational hardness assumptions. Their proofs are in the so-called random oracle model [2] where a hash function is treated as an ideal random function. We stress that although a proof in the random oracle model has a certain value it is still only a heuristic security argument for any implementation of the scheme. In particular, there exist cryptographic schemes that are provably secure in the random oracle model yet that are insecure with *any* possible standard-model instantiation of the hash function [12].

Details of our construction. In 1979 Rabin [35] proposed an encryption scheme based on the "modular squaring" trapdoor permutation whose one-wayness is equivalent to the factoring assumption. A semantically secure variant was later proposed by Goldwasser and Micali [22]. Our construction is based on the latter scheme [22] in its more efficient variant by Blum and Goldwasser [6] (which uses the Blum-Blum-Shub pseudorandom generator [5] to obtain an efficient hard-core function with linear output length). The Blum-Goldwasser scheme can easily be shown insecure against a CCA attack. Our main contribution consists of modifying the Blum-Goldwasser scheme such that it is provably CCA secure under the same hardness assumption yet it retains its high efficiency. Surprisingly, it is sufficient to add one additional group element to the ciphertexts that is then used for a consistency check in the decryption algorithm. For the consistency check itself, we also need to add two group elements to the public key.

Note that Paillier and Villar [32] (building on work of Williams [38]) show that the CCA security of schemes which only include an RSA modulus in the public key cannot be proven (using a black-box reduction) equivalent to factoring. In particular, this applies to the Blum-Goldwasser scheme [6] from which we start, so we have to modify the scheme's public key (and not only the ciphertexts). And indeed, given our modifications, our scheme's CCA security is *equivalent* to the factoring problem.

Proof Details. At a more technical level, the additional group elements in the public key can be set up by a simulator such that it is possible to decrypt (without the knowledge of the scheme's secret key) all consistent ciphertexts, except the ciphertext that is used to challenge the adversary. This "all-but-one" simulation technique can be traced back at least to [30], where it was used in the context of pseudorandom functions.[1] In the encryption context, "all-but-one" simulations have been used in identity-based encryption [8] and were already applied to several encryption schemes in [9,10,13,24,25].

The main novelty is that our proof makes direct use of the fact that the underlying primitive is a trapdoor one-way permutation, rather than the Diffie-Hellman problem. Therefore, the scheme's consistency check can be directly implemented by the simulator *without* having access to some external gap-oracle (as in [9,10,25]) or using other extrinsic rejection techniques (such as "hash proof systems" [15,14], "twinning" [13], or authenticated symmetric encryption [28,24][2]). Thus, our proof technique is fundamentally different from all known approaches

[1] We stress that our use of the term "all-but-one" refers to the ability to generate a secret key that can be used to decrypt all consistent ciphertexts except for an *externally given* ciphertext. This is very different from the techniques of, e.g., [31,16,15]: in these latter frameworks, the first step in the proof consists in *making the challenge ciphertext inconsistent*, and then constructing a secret key that can be used to construct *all* consistent ciphertexts. Hence, "all-but-one" really refers to an "artificially punctured" secret key.

[2] As opposed to generic CCA-secure symmetric encryption, a potentially weaker primitive.

to obtain CCA security. This also includes the recent class of schemes based on lossy trapdoor functions [33].

Efficiency. The resulting encryption scheme (which is actually a key encapsulation mechanism, see [15]) is very efficient: encryption needs roughly two, and decryption roughly one modular exponentiations; the public-key contains the modulus plus two group elements. (The modulus and one element can be viewed as systems parameters shared among all parties). To the best of our knowledge this is much more efficient than all known CCA-secure schemes based on an assumption *related* to factoring, even the ones based on a decisional assumption.

2 Preliminaries

2.1 Notation

We write $[N] = \{1, \ldots, N\}$. For group elements g, h, we denote by $\mathrm{dlog}_g h$ the discrete logarithm of h to the base g, i.e., the smallest $i \geq 0$ with $h = g^i$. A probabilistic polynomial-time (PPT) algorithm is a randomized algorithm which runs in strict polynomial time. If A is a probabilistic algorithm, we write $y \leftarrow A(x)$ to denote that the random variable y is defined as the output of A when run on input x and with fresh random coins. On the other hand, if S is a set, then $s \leftarrow S$ defines s as being uniformly and independently sampled from S. By k we denote the security parameter, which indicates the "amount of security" we desire. Typically, an adversarial advantage should be bounded by 2^{-k}, and a typical value for k is 80.

2.2 Factoring

A prime number P is called a *safe prime* iff $P = 2p + 1$ for a prime p. We assume a PPT algorithm IGen that, on input a security parameter k in unary, generates two random safe primes $P = 2p+1$ and $Q = 2q+1$ with $\mathsf{bitlength}(p) = \mathsf{bitlength}(q) = \ell_{\mathsf{N}}(k)/2 - 1$. We assume that p and q are odd, such that P and Q are congruent 3 modulo 4 and $N = PQ$ is a Blum integer. IGen returns N along with P and Q. Here $\ell_{\mathsf{N}}(k)$ denotes a function that represents, for any given security parameter k, the recommended (bit-)size of the composite modulus N. For the rest of the paper, we assume that N is generated by the factoring instance generator IGen. The set $\mathbb{QR}_N \subseteq \mathbb{Z}_N^*$ of *quadratic residues modulo* N is defined as $\mathbb{QR}_N := \{x \in \mathbb{Z}_N^* : \exists y \in \mathbb{Z}_N^* \text{ with } y^2 = x \bmod N\}$. Since $\mathbb{Z}_N^* \cong \mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_{pq}$, $\mathbb{QR}_N$ is a cyclic group of order pq. Note that this implies that a uniformly chosen element of $\mathbb{QR}_N$ is a generator (of $\mathbb{QR}_N$) with overwhelming probability. Computations in $\mathbb{QR}_N$ are computations modulo N. If it is implied by context, we omit writing explicitly "mod N" for calculations modulo N.

Definition 1 (Factoring assumption). *For an algorithm* F*, we define its* factoring advantage *as*

$$\mathsf{Adv}^{\mathsf{fac}}_{\mathsf{IGen},\mathsf{F}}(k) := \mathsf{Pr}\left[(N, P, Q) \leftarrow \mathsf{IGen}(1^k) \,:\, \mathsf{F}(N) = \{P, Q\}\right].$$

We say that F $(t_{\mathsf{fac}}, \epsilon_{\mathsf{fac}})$-factors composite integers *if* F *runs in time* t_{fac} *and* $\mathsf{Adv}^{\mathsf{fac}}_{\mathsf{IGen},\mathsf{F}}(k) \geq \epsilon(k)$*. The* factoring assumption *(with respect to* IGen*) states that* $\mathsf{Adv}^{\mathsf{fac}}_{\mathsf{IGen},\mathsf{F}}(k)$ *is negligible in* k *for every PPT* F*.*

The best algorithms currently known for factoring $N = PQ$ of length $\ell_{\mathsf{N}} = \mathsf{bitlength}(N) = \log N$ have (heuristic) running time

$$L_N(1/3, (64/9)^{1/3}) = e^{1.92 \ell_{\mathsf{N}}{}^{1/3+o(1)} (\log \ell_{\mathsf{N}})^{2/3}}.$$

Therefore, if we want k bits of security, we need to choose the function $\ell_{\mathsf{N}}(k)$ such that the above term is lower bounded by 2^k. As an example, one commonly uses $\ell_{\mathsf{N}}(80) = 1024$.

2.3 Key encapsulation mechanisms

Instead of a public-key encryption scheme we consider the conceptually simpler KEM framework. It is well-known that an $\mathsf{IND\text{-}CCA}$ secure KEM combined with a (one-time-)$\mathsf{IND\text{-}CCA}$ secure symmetric cipher (DEM) yields a $\mathsf{IND\text{-}CCA}$ secure public-key encryption scheme [15]. Efficient *one-time* $\mathsf{IND\text{-}CCA}$ secure DEMs can be constructed even without computational assumptions by using an encrypt-then-MAC paradigm [15] (or, alternatively, using computational assumptions such as strong pseudorandom permutations [34]).

A *key encapsulation mechanism (KEM)* $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three PPT algorithms. Via $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$, the key generation algorithm produces public/secret keys for security parameter $k \in \mathbb{N}$; via $(K, C) \leftarrow \mathsf{Enc}(pk)$, the encapsulation algorithm creates a symmetric key[3] $K \in \{0,1\}^{\ell_{\mathsf{K}}}$ together with a ciphertext C; via $K \leftarrow \mathsf{Dec}(sk, C)$, the possessor of secret key sk decrypts ciphertext C to get back a key K which is an element in $\{0,1\}^{\ell_{\mathsf{K}}}$ or a special reject symbol $\perp$. For correctness, we require that for all possible $k \in \mathbb{N}$, and all $(K, C) \leftarrow \mathsf{Enc}(pk)$, we have $\Pr[\mathsf{Dec}(sk, C) = K] = 1$, where the probability is taken over the choice of $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$, and the coins of all the algorithms in the expression above.

The common requirement for a KEM is indistinguishability against chosen-ciphertext attacks ($\mathsf{IND\text{-}CCA}$) [15], where an adversary is allowed to adaptively query a decapsulation oracle with ciphertexts to obtain the corresponding key. We are using the slightly simpler but equivalent one-phase definition from [26]. Formally:

Definition 2 ($\mathsf{IND\text{-}CCA}$ security of a KEM). *Let* $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a KEM. For any PPT algorithm* A*, we define the following experiments* $\mathsf{Exp}^{\mathsf{CCA\text{-}real}}_{\mathsf{KEM},\mathsf{A}}$ *and* $\mathsf{Exp}^{\mathsf{CCA\text{-}rand}}_{\mathsf{KEM},\mathsf{A}}$*:*

Experiment $\mathsf{Exp}^{\mathsf{CCA\text{-}real}}_{\mathsf{KEM},\mathsf{A}}(k)$	**Experiment** $\mathsf{Exp}^{\mathsf{CCA\text{-}rand}}_{\mathsf{KEM},\mathsf{A}}(k)$
$(pk, sk) \leftarrow \mathsf{Gen}(1^k)$	$(pk, sk) \leftarrow \mathsf{Gen}(1^k)$
	$R \leftarrow \{0,1\}^{\ell_{\mathsf{K}}}$
$(K^*, C^*) \leftarrow \mathsf{Enc}(pk)$	$(K^*, C^*) \leftarrow \mathsf{Enc}(pk)$
Return $\mathsf{A}^{\mathsf{Dec}(sk,\cdot)}(pk, K^*, C^*)$	*Return* $\mathsf{A}^{\mathsf{Dec}(sk,\cdot)}(pk, R, C^*)$

[3] For simplicity we assume that the KEM's keyspace are bitstrings of length ℓ_{K}.

In the above experiments, the decryption oracle $\mathsf{Dec}(sk,\cdot)$*, when queried with a ciphertext* $C \neq C^*$*, returns* $K \leftarrow \mathsf{Dec}(sk, C)$*. (*$\mathsf{Dec}(sk,\cdot)$ *ignores queries* $C = C^*$*.) We define* A*'s advantage in breaking* KEM*'s* $\mathsf{IND\text{-}CCA}$ *security as*

$$\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM},\mathsf{A}}(k) := \frac{1}{2}\left|\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{CCA\text{-}real}}_{\mathsf{KEM},\mathsf{A}}(k) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{CCA\text{-}rand}}_{\mathsf{KEM},\mathsf{A}}(k) = 1\right]\right|.$$

A $(t_{\mathsf{KEM}}, \epsilon_{\mathsf{KEM}})$-breaks KEM's $\mathsf{IND\text{-}CCA}$ security (short: A $(t_{\mathsf{KEM}}, \epsilon_{\mathsf{KEM}})$-breaks KEM) *if* A *runs in time at most* $t_{\mathsf{KEM}} = t_{\mathsf{KEM}}(k)$ *and we have* $\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM},\mathsf{A}}(k) \geq \epsilon_{\mathsf{KEM}}(k)$*. We say that* KEM has indistinguishable ciphertexts under chosen-ciphertext attacks (short: KEM is $\mathsf{IND\text{-}CCA}$ secure) *if for all PPT* A*, the function* $\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM},\mathsf{A}}(k)$ *is negligible in* k*.*

2.4 Target-collision resistant hashing

Informally, we say that a function $\mathsf{T} : X \to Y$ is a target-collision resistant (TCR) hash function (aka universal one-way hash function [31]), if, given a random preimage $x \in X$, it is hard to find $x' \neq x$ with $\mathsf{T}(x') = \mathsf{T}(x)$.

Definition 3 (TCR hash function). *Let* $\mathsf{T} : X \to Y$ *be a function. For an algorithm* B*, define*

$$\mathsf{Adv}^{\mathsf{TCR}}_{\mathsf{T},\mathsf{B}}(k) := \mathsf{Pr}\left[x \leftarrow X, x' \leftarrow \mathsf{B}(x) \,:\, x' \neq x \wedge \mathsf{T}(x') = \mathsf{T}(x)\right].$$

We say that B $(t_{\mathsf{T}}, \epsilon_{\mathsf{T}})$-breaks T's TCR property (short: B $(t_{\mathsf{T}}, \epsilon_{\mathsf{T}})$-breaks T) *iff* B*'s running time is at most* $t_{\mathsf{T}}(k)$ *and* $\mathsf{Adv}^{\mathsf{TCR}}_{\mathsf{T},\mathsf{B}}(k) \geq \epsilon_{\mathsf{T}}(k)$*. We say that* T *is* target-collision resistant *if for all PPT* B*, the function* $\mathsf{Adv}^{\mathsf{TCR}}_{\mathsf{T},\mathsf{B}}(k)$ *is negligible in* k*.*

3 Chosen-ciphertext security from factoring

3.1 The scheme

In this section, we will present our KEM construction. We will make use of two building blocks: a target collision-resistant hash function, and the Blum-Blum-Shub (BBS) pseudorandom number generator [5].

Concretely, for a product $N = PQ$ of two primes P, Q and $u \in \mathbb{Z}_N$, we establish the following notation: $|u|$ denotes the absolute value of u and $\mathsf{LSB}_N(u) = u \bmod 2$ the least significant bit of u, where in both cases u is interpreted as a signed integer with $-(N-1)/2 \leq u \leq (N-1)/2$. Furthermore, let

$$\mathsf{BBS}_N(u) = \left(\mathsf{LSB}_N(u), \mathsf{LSB}_N(u^2) \ldots, \mathsf{LSB}_N(u^{2^{\ell_{\mathsf{K}}-1}})\right) \in \{0,1\}^{\ell_{\mathsf{K}}}$$

denote the BBS generator applied to u and modulo N.[4]

[4] For efficiency, and at the price of a worse reduction, one can even simultaneously extract $\lceil \log_2 \log_2 N \rceil$ bits of each u^{2^i} instead of only the least significant bit [1]. However, our analysis treats the original BBS generator for simplicity.

Furthermore, for N as above, let $\mathsf{T} : \mathbb{Z}_N \to \{1, \ldots, 2^{\ell_{\mathsf{T}}} - 1\}$ be a target-collision resistant hash function.

The scheme. We are ready to define the following key encapsulation mechanism $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$:

Key generation. $\mathsf{Gen}(1^k)$ chooses uniformly at random

- a modulus $N = PQ = (2p+1)(2q+1)$ (using $\mathsf{IGen}(1^k)$, cf. Section 2.2),
- a quadratic residue $g \in \mathbb{QR}_N$,
- an exponent $\alpha \in [(N-1)/4]$,

Gen then sets $X = g^{\alpha 2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}}$ and outputs a public key pk and a secret key sk, where

$$pk = (N, g, X) \qquad\qquad sk = (N, g, \alpha).$$

Encapsulation. $\mathsf{Enc}(pk)$ chooses uniformly $r \in [(N-1)/4]$, sets

$$R = g^{r 2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}} \qquad t = \mathsf{T}(R) \in \{1, \ldots, 2^{\ell_{\mathsf{T}}} - 1\} \qquad S = \left|\left(g^t X\right)^r\right|$$

and outputs the key $K = \mathsf{BBS}_N(g^{r 2^{\ell_{\mathsf{T}}}}) \in \{0,1\}^{\ell_{\mathsf{K}}}$ and the ciphertext $C = (R, S) \in \mathbb{QR}_N \times (\mathbb{Z}_N^* \cap [(N-1)/2])$.

Decapsulation. $\mathsf{Dec}(sk, (R, S))$ verifies that $(R, S) \in \mathbb{Z}_N^* \times (\mathbb{Z}_N^* \cap [(N-1)/2])$ and rejects if not. Then, Dec computes $t = \mathsf{T}(R) \in \{1, \ldots, 2^{\ell_{\mathsf{T}}} - 1\}$, checks whether

$$\left(S^2\right)^{2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}} \stackrel{?}{=} \left(R^2\right)^{t + \alpha 2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}} \tag{1}$$

holds, and rejects if not. If (1) holds, Dec computes $a, b, c \in \mathbb{Z}$ such that

$$2^c = \gcd(t, 2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}) = at + b 2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}. \tag{2}$$

Note that $c < \ell_{\mathsf{T}}$ since $0 < t < 2^{\ell_{\mathsf{T}}}$. Then, Dec derives

$$T = \left(\left(S^2\right)^a \cdot \left(R^2\right)^{b - a\alpha}\right)^{2^{\ell_{\mathsf{T}} - c - 1}} \tag{3}$$

and from this $K = \mathsf{BBS}_N(T) \in \{0,1\}^{\ell_{\mathsf{K}}}$, which is the output.

We remark that decapsulation (or, rather, generation of the secret keys) does not require knowledge about the factorization of N. Indeed, the modulus N as well as the generator g can be viewed as global system parameters shared by many parties. Then pk only contains the value $X \in \mathbb{QR}_N$ and sk only contains $\alpha \in [(N-1)/4]$.

Our scheme uses an RSA modulus N that consists of safe primes. In Section 5 we show how to avoid this assumption and allow N to be an arbitrary Blum integer.

Correctness. The correctness of the scheme might not be obvious, so we prove it here. Fix a public key pk and a secret key sk as produced by $\mathsf{Gen}(1^k)$, and assume that (R, S) is a ciphertext for a key K as generated by $\mathsf{Enc}(pk)$. We have to show that $\mathsf{Dec}(sk, (R, S))$ outputs K. First, it is clear that $(R, S) \in \mathbb{Z}_N^* \times (\mathbb{Z}_N^* \cap [(N-1)/2])$. Also,

$$\left(S^2\right)^{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}} = \left(\left|\left(g^t X\right)^r\right|^2\right)^{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}} = g^{2(t+\alpha 2^{\ell_\mathsf{K}+\ell_\mathsf{T}})r2^{\ell_\mathsf{K}+\ell_\mathsf{T}}} \overset{(*)}{=} \left(R^2\right)^{t+\alpha 2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}$$

(where $(*)$ uses $R = g^{r2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}$), so (1) holds. Hence, (R, S) is not rejected by Dec. Now (1) implies

$$S^2 = \left(R^2\right)^{\frac{t+\alpha 2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}} = \left(R^2\right)^{\frac{t}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}+\alpha}, \tag{4}$$

where the division in the exponent is computed modulo $pq = |\mathbb{QR}_N|$. (Note that while S may or may not be a quadratic residue, S^2 certainly is.) This gives

$$\begin{aligned} T &\overset{(3)}{=} \left(\left(S^2\right)^a \cdot \left(R^2\right)^{b-a\alpha}\right)^{2^{\ell_\mathsf{T}-c-1}} = \left(\left(S^2 \cdot \left(R^2\right)^{-\alpha}\right)^a \cdot \left(R^2\right)^b\right)^{2^{\ell_\mathsf{T}-c-1}} \\ &\overset{(4)}{=} \left(\left(\left(R^2\right)^{\frac{t}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}}\right)^a \cdot \left(R^2\right)^b\right)^{2^{\ell_\mathsf{T}-c-1}} = \left(\left(R^2\right)^{\frac{at+b2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}}\right)^{2^{\ell_\mathsf{T}-c-1}} \\ &\overset{(2)}{=} \left(R^2\right)^{\frac{2^c}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}\cdot 2^{\ell_\mathsf{T}-c-1}} = \left(R^2\right)^{\frac{1}{2^{\ell_\mathsf{K}+1}}} \overset{(*)}{=} g^{r2^{\ell_\mathsf{T}}}, \end{aligned} \tag{5}$$

where, again, $(*)$ uses $R = g^{r2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}$. But (5) shows that Dec outputs $\mathsf{BBS}_N(T) = \mathsf{BBS}_N(g^{r2^{\ell_\mathsf{T}}}) = K$ as desired.

Theorem 1 (IND-CCA security of KEM). *Assume* T *is a target collision resistant hash function and the factoring assumption holds. Then* KEM *is* IND-CCA *secure in the sense of Definition 2.*

The proof of Theorem 1 will be given in Section 4.

Efficiency. We claim that, with some trivial optimizations, encapsulation uses roughly two exponentiations, and decapsulation roughly one exponentiation. Namely, encapsulation can first compute $A = g^r$ and $B = X^r$, which are two full exponentiations. Then, the remaining computations require only multiplications or exponentiations with very small exponents: $K = \mathsf{BBS}_N(A^{2^{\ell_\mathsf{T}}})$, $R = A^{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}$, and $S = A^t B$. (In fact, R is a by-product of computing K.) Similarly, decapsulation can first compute $D = R^\alpha / S$, which requires one full exponentiation. From D, (1) can be checked with $D^{2^{\ell_\mathsf{K}+\ell_\mathsf{T}+1}} \overset{?}{=} R^{2t}$, which requires only two exponentiations with very small exponents. The key K can then be computed as $\mathsf{BBS}_N(T)$ for $T = (R^b D^{-a})^{2^{\ell_\mathsf{T}-c}}$, which requires three exponentiations with small exponents (note that the bit-length of a and b is at most $\ell_\mathsf{K} + \ell_\mathsf{T}$).

For concreteness let us assume that one regular exponentiation with an exponent of length ℓ requires $1.5 \cdot \ell$ modular multiplications and that one squaring takes the same time as one multiplication. Let us further assume that $\ell_\mathsf{N} := \mathsf{bitlength}(N) = 1024$ and $\ell_\mathsf{K} = \ell_\mathsf{T} = 80$. Then encapsulation requires $3\ell_\mathsf{N} + \ell_\mathsf{K} + 2.5\ell_\mathsf{T} = 3352$ multiplications; decapsulation requires $1.5\ell_\mathsf{N} + 4\ell_\mathsf{K} + 6.5\ell_\mathsf{T} = 2376$ multiplications. In Appendix A we also propose a variant of our scheme that has slightly more efficient decapsulation but suffers from a comparatively large public key size.

We remark that, by adding the prime factors P and Q to the secret-key, we can further improve the scheme's efficiency. For example, using Chinese Remaindering will speed up decapsulation by a factor between 3 and 4.

4 Proof of security

We split up the proof of Theorem 1 into two parts:

- We first recall that the BBS generator is pseudorandom if factoring Blum integers is hard. This holds even if the modulus N and the $2^{\ell_{\mathsf{K}}}$-th power $u^{2^{\ell_{\mathsf{K}}}}$ of the BBS seed u are published, as is the case in our KEM. (Theorem 2.)
- We then prove that KEM is $\mathsf{IND\text{-}CCA}$ secure under the assumption that the BBS generator is pseudorandom and the employed hash function is target-collision resistant. This reduction is the heart of our proof. (Theorem 3.)

Combining both parts yields Theorem 1.

We start by recalling that the BBS generator is pseudorandom, in the following sense.

Definition 4 (PRNG experiment for BBS generator). *For an algorithm* D*, define*

$$\mathsf{Adv}^{\mathsf{BBS}}_{\mathsf{D}}(k) = \mathsf{Pr}\left[\mathsf{D}(N, z, \mathsf{BBS}_N(u)) = 1\right] - \mathsf{Pr}\left[\mathsf{D}(N, z, U_{\{0,1\}^{\ell_{\mathsf{K}}}}) = 1\right],$$

where

- $N \in \mathbb{N}$ *is distributed as* $\mathsf{IGen}(1^k)$*,*
- $u \in \mathbb{QR}_N$ *is uniformly chosen, and* $z = u^{2^{\ell_{\mathsf{K}}}}$*,*
- $U_{\{0,1\}^{\ell_{\mathsf{K}}}} \in \{0,1\}^{\ell_{\mathsf{K}}}$ *is independently and uniformly chosen.*

We say that D (t, ϵ)-breaks BBS *if* D*'s running time is at most* $t = t(k)$ *and* $\mathsf{Adv}^{\mathsf{BBS}}_{\mathsf{D}}(k) \geq \epsilon = \epsilon(k)$*.*

Concretely, any BBS-distinguisher can be used to factor Blum integers.

Theorem 2 (BBS-distinguisher $\Rightarrow$ factoring algorithm [7,5,1,18]). *For every algorithm* D *that* $(t_{\mathsf{BBS}}, \epsilon_{\mathsf{BBS}})$*-breaks* BBS*, there is an algorithm* F *that* $(t_{\mathsf{fac}}, \epsilon_{\mathsf{fac}})$*-factors Blum integers, where*

$$t_{\mathsf{fac}} \approx k^4 t_{\mathsf{BBS}}/\epsilon^2_{\mathsf{BBS}} \qquad\qquad \epsilon_{\mathsf{fac}} = \epsilon_{\mathsf{BBS}}/\ell_{\mathsf{K}}.$$

Proof. Let D be an algorithm that $(t_{\mathsf{BBS}}, \epsilon_{\mathsf{BBS}})$-breaks BBS. [7] show that D gives rise to an algorithm D' that $(t_{\mathsf{LSB}}, \epsilon_{\mathsf{LSB}})$-distinguishes tuples $(N, u^2, \mathsf{LSB}(u))$ from tuples $(N, u^2, U_{\{0,1\}})$, where $u \in \mathbb{QR}_N$ and $U_{\{0,1\}} \in \{0,1\}$ are uniformly chosen, $t_{\mathsf{LSB}} \approx t_{\mathsf{BBS}}$, and $\epsilon_{\mathsf{LSB}} = \epsilon_{\mathsf{BBS}}/\ell_{\mathsf{K}}$. Building on [1], [18] show how to transform D' into an algorithm F that $(t_{\mathsf{fac}}, \epsilon_{\mathsf{fac}})$-factors Blum integers, where $t_{\mathsf{fac}} \approx k^2 t_{\mathsf{LSB}}/\epsilon^2_{\mathsf{LSB}} \approx k^4 t_{\mathsf{BBS}}/\epsilon^2_{\mathsf{BBS}}$ and $\epsilon_{\mathsf{fac}} = \epsilon_{\mathsf{LSB}} = \epsilon_{\mathsf{BBS}}/\ell_{\mathsf{K}}$. (We use the interpretation [30, Theorem 6.1] of the results from [18] here.) The claim follows.

The following theorem contains the heart of our proof, namely, a simulation that shows that any successful $\mathsf{IND\text{-}CCA}$ adversary on KEM implies a successful BBS-distinguisher (and hence, using Theorem 2, can be used to factor Blum integers).

Theorem 3 ($\mathsf{IND\text{-}CCA}$ adversary ⇒ BBS-distinguisher). *For every adversary A that $(t_{\mathsf{KEM}}, \epsilon_{\mathsf{KEM}})$-breaks KEM's $\mathsf{IND\text{-}CCA}$ property, there exists an algorithm D that $(t_{\mathsf{BBS}}, \epsilon_{\mathsf{BBS}})$ breaks BBS and an adversary B that $(t_{\mathsf{T}}, \epsilon_{\mathsf{T}})$-breaks T, such that*

$$t_{\mathsf{BBS}} \approx t_{\mathsf{T}} \approx t_{\mathsf{KEM}} \qquad\qquad \epsilon_{\mathsf{BBS}} + \epsilon_{\mathsf{T}} + 2^{-k+3} \geq \epsilon_{\mathsf{KEM}}.$$

Proof. **Setting up the variables for simulation.** Assume an adversary A on KEM's $\mathsf{IND\text{-}CCA}$ security. We define a BBS-distinguisher D, which acts on input (N, z, V) as follows. D first uniformly selects a quadratic residue $g \in \mathbb{QR}_N$, as well as exponent $\beta \in [(N-1)/4]$, and sets

$$R^* = z \qquad t^* = \mathsf{T}(R^*) \in \{1, \ldots, 2^{\ell_{\mathsf{T}}} - 1\} \qquad X = g^{\beta 2^{\ell_{\mathsf{K}} + \ell_{\mathsf{T}}} - t^*}.$$

The public key used in the simulation is $pk = (N, g, X)$. It will be convenient to write $X = g^{\alpha 2^{\ell_{\mathsf{K}} + \ell_{\mathsf{T}}}}$ as in Gen, for $\alpha = \beta - t^*/2^{\ell_{\mathsf{K}} + \ell_{\mathsf{T}}}$ unknown to D. (Here and in the following, a division of exponents is computed modulo pq, the order of $\mathbb{QR}_N$.) Furthermore, in the following, we will silently assume that g generates $\mathbb{QR}_N$, which is very likely, but not guaranteed. A rigorous justification that takes into account error probabilities follows below.

Preparation of challenge ciphertext and key. To complete the definition of the challenge ciphertext $C^* = (R^*, S^*)$, write $R^* = g^{2^{\ell_{\mathsf{K}} + \ell_{\mathsf{T}}} r^*}$. Since we assumed that g is a generator, this is possible, but of course r^* is unknown. D defines

$$S^* = \left| R^{*\beta} \right| \quad \left(= \left| g^{r^* \beta 2^{\ell_{\mathsf{K}} + \ell_{\mathsf{T}}}} \right| = \left| \left(g^{t^*} X \right)^{r^*} \right| \right) \tag{6}$$

as Enc would have computed. The (real) corresponding key K^* is defined as

$$K^* = \mathsf{BBS}_N \left(g^{2^{\ell_{\mathsf{T}}} r^*} \right) = \mathsf{BBS}_N \left(R^{* \frac{1}{2^{\ell_{\mathsf{K}}}}} \right) = \mathsf{BBS}_N \left(z^{\frac{1}{2^{\ell_{\mathsf{K}}}}} \right) = \mathsf{BBS}_N \left(u \right) \ . \tag{7}$$

D then invokes A with public key $pk = (N, g, X)$, challenge ciphertext $C^* = (R^*, S^*)$, and challenge key V. Note that V is either the real challenge key $\mathsf{BBS}_N(u)$, or it is a uniform string.

On the distribution of simulated public key and challenge ciphertext. We claim that the distribution of public key pk and challenge ciphertext C^* is almost identical in simulation and $\mathsf{IND\text{-}CCA}$ experiment. Concretely, we postpone the straightforward but somewhat tedious proof of the following lemma until after the description of our simulation.

Lemma 1. *There exists an event* $\mathsf{bad}_{\mathsf{key}}$ *such that, conditioned on* $\neg\mathsf{bad}_{\mathsf{key}}$*, public key pk and challenge ciphertext* C^* *are identically distributed in simulation and* IND-CCA *experiment. Also,* $\neg\mathsf{bad}_{\mathsf{key}}$ *implies that g is a generator. We have*

$$\mathsf{Pr}\left[\mathsf{bad}_{\mathsf{key}}\right] \leq 2^{-k+3} \tag{8}$$

both in the simulation and in the IND-CCA *experiment.*

Thus, conditioned on $\neg\mathsf{bad}_{\mathsf{key}}$, D perfectly simulates A's input as in the real IND-CCA experiment if $V = \mathsf{BBS}_N(u) = \mathsf{BBS}_N(z^{1/2^{\ell_{\mathsf{K}}}})$, and as in the ideal IND-CCA experiment if V is random.

How to handle A's decryption queries. It remains to describe how D handles decryption queries of A as in the IND-CCA experiment. So say that A submits a ciphertext (R, S) for decryption. We may assume that $(R, S) \in \mathbb{Z}_N^* \times (\mathbb{Z}_N^* \cap [(N-1)/2])$. Let $t = \mathsf{T}(R) \in \{1, \ldots, 2^{\ell_{\mathsf{T}}} - 1\}$. We call a ciphertext *consistent* iff the original decryption algorithm would not have rejected it. Hence, by (1), a ciphertext is consistent iff

$$\left(S^2\right)^{2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}} \overset{?}{=} \left(R^2\right)^{t-t^*+\beta 2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}} \quad \left(= \left(R^2\right)^{t+\alpha 2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}}\right) . \tag{9}$$

By our setup of variables, D can check (9) by itself, and hence detect and reject inconsistent ciphertexts.

How to decrypt consistent ciphertexts. Now assume that C is consistent and $t \neq t^*$. Then, (4) and (5) follow (except for the deduction $(*)$) just as in the correctness proof, and we get

$$T = \left(R^2\right)^{\frac{1}{2^{\ell_{\mathsf{K}}+1}}} \tag{10}$$

for the raw key T that would have been computed by Dec. We will now show how D can compute T. Namely, D computes $a', b', c' \in \mathbb{Z}$ such that

$$2^{c'} = \gcd(t - t^*, 2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}) = a'(t - t^*) + b' 2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}. \tag{11}$$

Since $1 \leq t, t^* < 2^{\ell_{\mathsf{T}}}$ and $t \neq t^*$, we have $c' < \ell_{\mathsf{T}}$. Similarly to (4) and (5), we obtain

$$S^2 = \left(R^2\right)^{\frac{t-t^*}{2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}}+\beta}, \tag{12}$$

from (9), and from this

$$\begin{aligned}
\left(\left(S^2\right)^{a'} \cdot \left(R^2\right)^{b'-a'\beta}\right)^{2^{\ell_{\mathsf{T}}-c'-1}} &= \left(\left(S^2 \cdot \left(R^2\right)^{-\beta}\right)^{a'} \cdot \left(R^2\right)^{b'}\right)^{2^{\ell_{\mathsf{T}}-c'-1}} \\
\overset{(12)}{=} \left(\left(\left(R^2\right)^{\frac{t-t^*}{2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}}}\right)^{a'} \cdot \left(R^2\right)^{b'}\right)^{2^{\ell_{\mathsf{T}}-c'-1}} &= \left(\left(R^2\right)^{\frac{a'(t-t^*)+b'2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}}{2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}}}\right)^{2^{\ell_{\mathsf{T}}-c'-1}} \\
\overset{(11)}{=} \left(R^2\right)^{\frac{2^{c'}}{2^{\ell_{\mathsf{K}}+\ell_{\mathsf{T}}}} \cdot 2^{\ell_{\mathsf{T}}-c'-1}} &= \left(R^2\right)^{\frac{1}{2^{\ell_{\mathsf{K}}+1}}} \overset{(10)}{=} T.
\end{aligned} \tag{13}$$

Note that from T, the final decryption key can be computed as $K = \mathsf{BBS}_N(T)$. Hence, using (13), D can correctly decrypt every consistent ciphertext with $t \neq t^*$.

The case $t = t^*$. So let us turn to the case that $t = t^*$ and the ciphertext is consistent. Then, if $R = R^*$ holds, we have

$$S^2 \overset{(9)}{=} \left(R^2\right)^{\frac{t-t^*}{2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}+\beta} \overset{(*)}{=} \left(R^{*2}\right)^\beta = S^{*2} \tag{14}$$

where in $(*)$ we use $R = R^*$ and $t = t^*$. Furthermore, $(R, S) \neq (R^*, S^*)$ implies $|S| = S \neq S^* = |S^*|$, so that $S \neq \pm S^*$ and $(S + S^*)(S - S^*) = S^2 - S^{*2} \overset{(14)}{=} 0 \bmod N$ yields a non-trivial factorization of N. Hence, D can efficiently factor N to solve its own input challenge (N, z, L) directly whenever $R = R^*$ and (R, S) is consistent.

On the other hand, if $\mathsf{T}(R) = t = t^* = \mathsf{T}(R^*)$ and $R \neq R^*$, then A has broken the target-collision resistance of T. Formally, let $\mathsf{bad}_\mathsf{TCR}$ denote the event that $t = t^*$ and $R \neq R^*$. If $\mathsf{bad}_\mathsf{TCR}$ occurs, D can safely give up, since

$$\mathsf{Pr}\left[\mathsf{bad}_\mathsf{TCR}\right] \leq \mathsf{Adv}^\mathsf{TCR}_{\mathsf{T},\mathsf{B}}(k) \tag{15}$$

for a suitable PPT adversary B on T that simulates D and A.

Summary of the decryption procedure. We summarize the decryption cases:

- inconsistent (R, S) (consistency check (9)$\Leftrightarrow$(1) not passed): reject,
- consistent (R, S) and $t \neq t^*$: decrypt using (13),
- consistent (R, S), $t = t^*$, and $R = R^*$: factor N (using $S \neq \pm S^*$ and $S^2 = S^{*2}$ by (14)),
- consistent (R, S), $t = t^*$, and $R \neq R^*$: give up simulation (A has found a T-collision).

Hence, also decryption is faithfully simulated unless $\mathsf{bad}_\mathsf{TCR}$ occurs.

Finishing the proof. We conclude that, unless $\mathsf{bad}_\mathsf{TCR}$ or $\mathsf{bad}_\mathsf{key}$ occurs, D perfectly simulates the real IND-CCA experiment upon input $V = \mathsf{BBS}_N(u)$, and the ideal IND-CCA experiment if V is random. If we let D output whatever the simulated experiment outputs, we obtain:

$$\begin{aligned}
&\left|\mathsf{Pr}\left[\mathsf{D}(N, z, \mathsf{BBS}_N(u)) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}^{\text{CCA-real}}_{\mathsf{KEM},\mathsf{A}}(k) = 1\right]\right| \leq \mathsf{Pr}\left[\mathsf{bad}_\mathsf{TCR}\right] + \mathsf{Pr}\left[\mathsf{bad}_\mathsf{key}\right] \\
&\left|\mathsf{Pr}\left[\mathsf{D}(N, z, U_{\{0,1\}^{\ell_\mathsf{K}}}) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}^{\text{CCA-rand}}_{\mathsf{KEM},\mathsf{A}}(k) = 1\right]\right| \leq \mathsf{Pr}\left[\mathsf{bad}_\mathsf{TCR}\right] + \mathsf{Pr}\left[\mathsf{bad}_\mathsf{key}\right].
\end{aligned} \tag{16}$$

Using (8) and (15), Theorem 3 follows from (16).

It remains to prove Lemma 1.

Proof of Lemma 1. Observe that pk and C^* are distributed slightly differently in the IND-CCA experiment (i.e., as generated by Gen and Enc) and in the simulation:

- $R^* = g^{r^*}$ for uniform (hidden) $r^* \in [(N-1)/4]$ in the experiment, while $R^* \in \mathbb{QR}_N$ is a uniform group element in the simulation.
- $X = g^{\alpha 2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}$ for uniform (hidden) $\alpha \in [(N-1)/4]$ in the experiment, while $X = g^{\beta 2^{\ell_\mathsf{K}+\ell_\mathsf{T}} - t^*}$ for uniform (hidden) $\beta \in [(N-1)/4]$ in the simulation.

However, conditioned on the following event $\mathsf{good}_\mathsf{key}$:

(in the experiment:) g is a generator, and $r^*, \alpha \leq |\mathbb{QR}_N|$,

(in the simulation:) g is a generator, and $\beta \leq |\mathbb{QR}_N|$,

pk and C^* are distributed identically in experiment and simulation: $\mathsf{good}_\mathsf{key}$ implies that N, g, X, and R^* are uniformly and independently chosen over their respective domains, and S^* follows deterministically from pk and R^* according to (7). Hence we only need to bound the probability of $\mathsf{bad}_\mathsf{key} := \neg\mathsf{good}_\mathsf{key}$. Since $|\mathbb{QR}_N| = pq$ and we assumed that p and q are $n/2$-bit primes, a uniform $\mathbb{QR}_N$-element is a generator except with probability $(p+q-1)/pq \leq 2^{-n/2+2}$. Furthermore, $(N-1)/4$ is a close approximation of the group order $|\mathbb{QR}_N| = pq = (N-1)/4 - (p+q)/2$, so that, e.g., $r^* \leq |\mathbb{QR}_N|$ except with probability $2(p+q)/(N-1) \leq 2^{-n/2+1}$. Hence,

$$\Pr\left[\mathsf{bad}_\mathsf{key}\right] \leq \max\left\{2^{-n/2+2} + 2\cdot 2^{-n/2+1}, 2^{-n/2+2} + 2^{-n/2+1}\right\}$$
$$= 2^{-n/2+3} \overset{n/2\geq k}{\leq} 2^{-k+3}$$

both in the experiment and in the simulation.

5 Avoiding safe primes

In our KEM, we assume that $N = PQ$ is composed of two safe primes (i.e., primes of the form $P = 2p+1$ for prime p). We can drop this assumption and allow arbitrary Blum integers N, if we employ a Goldreich-Levin [21] based pseudorandom generator instead of the Blum-Blum-Shub generator. Namely, all we actually need to prove that KEM is $\mathsf{IND\text{-}CCA}$ is that

$$\left(N, g, g^{r2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}, \mathsf{Ext}_{pk}(g^{r2^{\ell_\mathsf{T}}})\right) \overset{\mathrm{c}}{\approx} \left(N, g, g^{r2^{\ell_\mathsf{K}+\ell_\mathsf{T}}}, U_{\{0,1\}^{\ell_\mathsf{K}}}\right), \tag{17}$$

where $\overset{\mathrm{c}}{\approx}$ denotes computational indistinguishability, N is a Blum integer, $g \in \mathbb{QR}_N$, $r \in [N/4]$, and $U_{\{0,1\}^{\ell_\mathsf{K}}} \in \{0,1\}^{\ell_\mathsf{K}}$ are uniform, and Ext is a suitable randomness extractor. In our original description of KEM, we have $\mathsf{Ext}_{pk}(u) = \mathsf{BBS}_N(u)$. In that case, we only know that the hardness of factoring N implies (17) if $u = g^{r2^{\ell_\mathsf{T}}}$ is a *uniform* element of $\mathbb{QR}_N$ (which is the case when $N = PQ$ for safe primes P, Q, since then g is a generator with high probability). But if g is not a generator at least with high probability, then u may not be uniformly distributed.

Now suppose we set

$$\mathsf{Ext}_{pk}(u) = \left(\mathsf{GL}_s(u), \mathsf{GL}_s(u^2), \ldots, \mathsf{GL}_s(u^{2^{\ell_\mathsf{K}-1}})\right) \in \{0,1\}^{\ell_\mathsf{K}}$$

for the Goldreich-Levin predicate GL_s that maps u to the bitwise inner product of s and u. Then a hybrid argument and the hard-core property of GL_s show that (17) is implied by the hardness of computing u with $u^2 = v \mod N$ from (N, g, v) (with $v = g^r$). But any algorithm B that computes such a u from (N, g, v) can be used to factor N. Namely, given N, choose uniformly $h \in \mathbb{Z}_N$ and $\tilde{r} \in [N/4]$, and set $g = h^2$ and $v = g^{2\tilde{r}+1}$. (Observe that v is almost uniformly distributed over $\langle g \rangle$, since N is a Blum integer.) Then, invoke $B(N, g, v)$ to obtain a square root u of v. We can then compute a square root of g as $\tilde{h} = u^a g^b$ (for $a, b \in \mathbb{Z}$ with $a(2\tilde{r} + 1) + 2b = \gcd(2\tilde{r} + 1, 2) = 1$). With probability 1/2, then $\gcd(h - \tilde{h}, N)$ yields a non-trivial factor of N. Hence (17) is implied by the hardness of factoring arbitrary Blum integers, and our KEM (instantiated with the Goldreich-Levin predicate) is $\mathsf{IND\text{-}CCA}$ secure. The price to pay is that we need to place a seed $s \in \{0,1\}^{\ell_\mathsf{N}}$ for the Goldreich-Levin hard-core function in the public key. (However, note that s can be made a global system parameter, like N and g.)

Acknowledgements

We are grateful to Victor Shoup, who generously allowed us to use his observations on how to compress the public key from $O(\ell_\mathsf{T})$ down to two group elements, and on how to get rid of the assumption that P and Q are safe primes. We would also like to thank Ronald Cramer and Ivan Damgård for interesting discussions.

References

1. Werner Alexi, Benny Chor, Oded Goldreich, and Claus-Peter Schnorr. RSA and Rabin functions: Certain parts are as hard as the whole. *SIAM Journal on Computing*, 17(2):194–209, 1988.
2. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.
3. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111. Springer-Verlag, Berlin, Germany, May 1994.
4. Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 1–12. Springer-Verlag, Berlin, Germany, August 1998.
5. Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, May 1986.
6. Manuel Blum and Shafi Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 289–302. Springer-Verlag, Berlin, Germany, August 1985.
7. Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

8. Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer-Verlag, Berlin, Germany, May 2004.
9. Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):915–942, 2006.
10. Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 05*, pages 320–329. ACM Press, November 2005.
11. Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144. Springer-Verlag, Berlin, Germany, August 2003.
12. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, 2004.
13. David Cash, Eike Kiltz, and Victor Shoup. The Twin Diffie-Hellman problem and applications. In Nigel P. Smart, editor, *EUROCRYPT 2008*, LNCS, pages 127–145. Springer-Verlag, Berlin, Germany, April 2008.
14. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer-Verlag, Berlin, Germany, April / May 2002.
15. Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.
16. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991.
17. Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.
18. Roger Fischlin and Claus-Peter Schnorr. Stronger security proofs for RSA and Rabin bits. *Journal of Cryptology*, 13(2):221–244, 2000.
19. Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. *ACM Transactions on Information and System Security*, 9(2):181–234, 2006.
20. Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.
21. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.
22. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
23. Goichiro Hanaoka and Kaoru Kurosawa. Efficient chosen ciphertext secure public key encryption under the computational Diffie-Hellman assumption. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, LNCS, pages 308–325. Springer, December 2008.
24. Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 553–571. Springer-Verlag, Berlin, Germany, August 2007.
25. Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 581–600. Springer-Verlag, Berlin, Germany, March 2006.

26. Eike Kiltz. Chosen-ciphertext secure key-encapsulation based on Gap Hashed Diffie-Hellman. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 282–297. Springer-Verlag, Berlin, Germany, April 2007.
27. Eike Kiltz, Krzysztof Pietrzak, Martijn Stam, and Moti Yung. A new randomness extraction paradigm for hybrid encryption. In Antoine Joux, editor, *EUROCRYPT 2009*, LNCS, pages ???–??? Springer, 2009.
28. Kaoru Kurosawa and Yvo Desmedt. A new paradigm of hybrid encryption scheme. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 426–442. Springer-Verlag, Berlin, Germany, August 2004.
29. Stefan Lucks. A variant of the Cramer-Shoup cryptosystem for groups of unknown order. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 27–45. Springer-Verlag, Berlin, Germany, December 2002.
30. Moni Naor, Omer Reingold, and Alon Rosen. Pseudo-random functions and factoring. *SIAM Journal on Computing*, 31(5):1383–1404, 2002.
31. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*. ACM Press, May 1990.
32. Pascal Paillier and Jorge L. Villar. Trading one-wayness against chosen-ciphertext security in factoring-based encryption. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 252–266. Springer-Verlag, Berlin, Germany, December 2006.
33. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 187–196. ACM Press, May 2008.
34. Duong Hieu Phan and David Pointcheval. About the security of ciphers (semantic security and pseudo-random permutations). In Helena Handschuh and Anwar Hasan, editors, *SAC 2004*, volume 3357 of *LNCS*, pages 182–197. Springer-Verlag, Berlin, Germany, August 2004.
35. Michael O. Rabin. Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, January 1979.
36. Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer-Verlag, Berlin, Germany, August 1992.
37. Leonid Reyzin and Natan Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In Lynn Margaret Batten and Jennifer Seberry, editors, *Information Security and Privacy, 7th Australian Conference, ACISP 2002*, LNCS, pages 144–154, Melbourne, Australia, July 2002. Springer.
38. Hugh C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory*, 26(6):726–729, November 1980.

A A variant of our scheme

We now propose a variant of our scheme that has slightly more efficient decapsulation but suffers from a comparatively large public key size.

Let $\mathsf{T} : \mathbb{Z}_N \to \{0,1\}^{\ell_\mathsf{T}}$ be a target-collision resistant hash function. Then, define the following key encapsulation mechanism $\mathsf{KEM}' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$:
Key generation. $\mathsf{Gen}'(1^k)$ chooses uniformly at random

- a modulus $N = PQ = (2p+1)(2q+1)$ (using $\mathsf{IGen}(1^k)$, cf. Section 2.2),
- a quadratic residue $h \in \mathbb{QR}_N$,
- exponents $\alpha_{i,j} \in [(N-1)/4]$ for $i \in [\ell_\mathsf{T}]$ and $j \in \{0,1\}$,

Gen' then gets $g = h^2$ and

$$\begin{aligned} X_{1,j} &= g^{\alpha_{1,j}2^{\ell_\mathsf{K}}} \cdot h && j \in \{0,1\} \\ X_{i,j} &= g^{\alpha_{i,j}2^{\ell_\mathsf{K}}} & i = 2, \dots, \ell_\mathsf{T}, \quad & j \in \{0,1\}. \end{aligned}$$

Gen' finally outputs a public key pk and a secret key sk, where

$$pk = (N, g, (X_{i,j})_{i \in [\ell_\mathsf{T}], j \in \{0,1\}}) \qquad sk = (N, (\alpha_{i,j})_{i \in [\ell_\mathsf{T}], j \in \{0,1\}}).$$

Encapsulation. $\mathsf{Enc}'(pk)$ chooses uniformly $r \in [(N-1)/4]$, sets

$$R = g^{2^{\ell_\mathsf{K}} r} \qquad t = (t_1, \dots, t_{\ell_\mathsf{T}}) = \mathsf{T}(R) \in \{0,1\}^{\ell_\mathsf{T}} \qquad S = \left| \left(\prod_{i=1}^{\ell_\mathsf{T}} X_{i,t_i} \right)^r \right|$$

and outputs the key $K = \mathsf{BBS}_N(g^r) \in \{0,1\}^{\ell_\mathsf{K}}$ and the ciphertext $C = (R,S) \in \mathbb{QR}_N \times (\mathbb{Z}_N^* \cap [(N-1)/2])$.

Decapsulation. $\mathsf{Dec}'(sk, (R,S))$ verifies that $(R,S) \in \mathbb{Z}_N^* \times (\mathbb{Z}_N^* \cap [(N-1)/2])$ and rejects if not. Then, Dec' computes $t = (t_1, \dots, t_{\ell_\mathsf{T}}) = \mathsf{T}(R)$ and checks

$$\left(S^2\right)^{2^{\ell_\mathsf{K}+1}} \overset{?}{=} \left(R^2\right)^{1+2^{\ell_\mathsf{K}+1}\sum_{i=1}^{\ell_\mathsf{T}} \alpha_{i,t_i}} \qquad \left(= \left(R^2\right)^{2\mathrm{dlog}_g \prod_{i=1}^{\ell_\mathsf{T}} X_{i,t_i}} \right). \tag{18}$$

If (18) does not hold, Dec' rejects. Otherwise, Dec' computes

$$T = S^2 \left(R^2\right)^{-\sum_{i=1}^{\ell_\mathsf{T}} \alpha_{i,t_i}} \qquad \left(\overset{(18)}{=} \left(R^2\right)^{\frac{1}{2^{\ell_\mathsf{K}+1}}} \right). \tag{19}$$

and outputs $K = \mathsf{BBS}_N(T) = \mathsf{BBS}_N((R^2)^{\frac{1}{2^{\ell_\mathsf{K}+1}}})$.

Correctness. The scheme enjoys correctness, since an honestly generated ciphertext (R,S) fulfils

$$\begin{aligned} \left(S^2\right)^{2^{\ell_\mathsf{K}+1}} &= \left(\left| \left(\prod_{i=1}^{\ell_\mathsf{T}} X_{i,t_i} \right)^r \right|^2 \right)^{2^{\ell_\mathsf{K}+1}} \\ &= g^{2^{\ell_\mathsf{K}+1} r \cdot 2\mathrm{dlog}_g \prod_{i=1}^{\ell_\mathsf{T}} X_{i,t_i}} = \left(R^2\right)^{2\mathrm{dlog}_g \prod_{i=1}^{\ell_\mathsf{T}} X_{i,t_i}}, \end{aligned}$$

so that consistency in the sense of (18) holds, and $\mathsf{Dec}'(sk,(R,S))$ outputs $\mathsf{BBS}_N(T) = \mathsf{BBS}_N((R^2)^{\frac{1}{2^{\ell_\mathsf{K}+1}}}) = \mathsf{BBS}_N(g^r) = K$.

Efficiency. With some trivial optimizations, KEM' uses roughly two exponentiations for encapsulation and one for decapsulation. However, KEM''s public key contains $2\ell_\mathsf{T} + 1$ group elements. This number can be roughly halved by using the following technique. Namely, observe that in KEM', the hash value $t = \mathsf{T}(R)$ is interpreted as a *bitwise* selector of ℓ_{T} out of $2\ell_{\mathsf{T}}$ group elements $X_{i,j}$. Instead, one can interpret t as an *integer* that specifies a subset of group elements X_i. Concretely, we can define a mapping f from $\{0,1\}^{\ell_{\mathsf{T}}}$ to subsets of $[\ell]$, where ℓ denotes the number of group elements. For our proof, we will only need that for any distinct $t, t^* \in \{0,1\}^{\ell_{\mathsf{T}}}$, we have $f(t) \not\subseteq f(t^*)$ and $f(t^*) \not\subseteq f(t)$. As an example, we can have the injective mapping f that associates to t the t-th subset of $[\ell]$ of size $\ell/2$. Using a suitable enumeration of these subsets, f can be implemented efficiently, and we will only need about $\ell \approx \ell_{\mathsf{T}} + \log \ell_{\mathsf{T}}$ group elements X_i to make f injective. More sophisticated examples of such mappings f were suggested in the context of one-time signatures, see [37]. However, since our scheme KEM is far superior in public key size and (roughly) on par with KEM' in terms of efficiency, we omit the details.

For concreteness let us again assume that one regular exponentiation with an exponent of length ℓ requires $1.5 \cdot \ell$ modular multiplications and that one squaring takes the same time as one multiplication. Let us further assume that $\ell_{\mathsf{N}} := \mathsf{bitlength}(N) = 1024$ and $\ell_{\mathsf{K}} = \ell_{\mathsf{T}} = 80$. Then encapsulation requires $3\ell_{\mathsf{N}} + \ell_{\mathsf{K}} + \ell_{\mathsf{T}} = 3232$ multiplications; decapsulation requires $1.5\ell_{\mathsf{N}} + \ell_{\mathsf{K}} = 1616$ multiplications. Hence in terms of efficiency KEM' is slightly better than KEM, in particular for decapsulation. However, KEM' has the drawback of large public key size.

Theorem 4 **($\mathsf{IND\text{-}CCA}$ security of KEM').** *Assume* T *is a target collision resistant hash function and the factoring assumption holds. Then* KEM' *is secure in the* $\mathsf{IND\text{-}CCA}$ *secure in the sense of Definition 2.*

Given Theorem 2, it suffices to prove the following theorem.

Theorem 5 **($\mathsf{IND\text{-}CCA}$ adversary on $\mathsf{KEM}' \Rightarrow \mathsf{BBS}$-distinguisher).** *For every adversary* A *that* $(t_{\mathsf{KEM}'}, \epsilon_{\mathsf{KEM}'})$*-breaks* KEM'*'s* $\mathsf{IND\text{-}CCA}$ *property, there exists an algorithm* D *that* $(t_{\mathsf{BBS}}, \epsilon_{\mathsf{BBS}})$ *breaks* BBS *and an adversary* B *that* $(t_{\mathsf{T}}, \epsilon_{\mathsf{T}})$*-breaks* T*, such that*

$$t_{\mathsf{BBS}} \approx t_{\mathsf{T}} \approx t_{\mathsf{KEM}'} \qquad\qquad \epsilon_{\mathsf{BBS}} + \epsilon_{\mathsf{T}} + \frac{2\ell_{\mathsf{T}} + 3}{2^{k-1}} \geq \epsilon_{\mathsf{KEM}'}.$$

Proof. **Setting up the variables for simulation.** Assume an adversary A on KEM''s $\mathsf{IND\text{-}CCA}$ security. We define a BBS-distinguisher D, which acts on input (N, z, V) as follows. D first uniformly picks an element $h \in \mathbb{QR}_N$, exponents $\beta_{i,j} \in [(N-1)/4]$ for $i \in [\ell_{\mathsf{T}}]$, $j \in \{0,1\}$, and sets

$$L = \mathrm{lcm}(1, \ldots, \ell_{\mathsf{T}}) \qquad g = h^{2L} \qquad R^* = z$$
$$t^* = (t^*_1, \ldots, t^*_{\ell_{\mathsf{T}}}) = \mathsf{T}(R^*) \qquad X_{i,t^*_i} = g^{\beta_{i,t^*_i} 2^{\ell_{\mathsf{K}}}} \qquad X_{i,1-t^*_i} = g^{\beta_{i,1-t^*_i} 2^{\ell_{\mathsf{K}}}} \cdot h.$$

Preparation of challenge ciphertext and key. To complete the definition of the challenge ciphertext $C^* = (R^*, S^*)$, D defines

$$S^* = \left| R^{*\sum_{i=1}^{\ell_{\mathsf{T}}} \beta_{i,t^*_i}} \right| \qquad \left(= \left| R^{*\frac{\mathrm{dlog}_g \prod_{i=1}^{\ell_{\mathsf{T}}} X_{i,t^*_i}}{2^{\ell_{\mathsf{K}}}}} \right| \right) \tag{20}$$

such that (18) is met. Note that the (real) corresponding key K^* according to Dec' is defined as

$$K^* = \mathsf{BBS}_N\left(\left(R^{*2}\right)^{\frac{1}{2^{\ell_\mathsf{K}+1}}}\right) = \mathsf{BBS}_N(z^{\frac{1}{2^{\ell_\mathsf{K}}}}) = \mathsf{BBS}_N(u). \tag{21}$$

D then invokes A with public key $pk = (N, g, (X_{i,j})_{i,j})$, challenge ciphertext $C^* = (R^*, S^*)$, and challenge key V. Note that V is either the real challenge key $\mathsf{BBS}_N(u)$, or it is a uniform string.

On the distribution of simulated public key and challenge ciphertext. We claim that the distribution of public key pk and challenge ciphertext C^* is almost identical in simulation and $\mathsf{IND\text{-}CCA}$ experiment. Concretely, the proof of the following lemma is very similar to the proof of Theorem 1, and we omit it.

Lemma 2. *There exists an event* $\mathsf{bad}_\mathsf{key}$ *such that, conditioned on* $\neg\mathsf{bad}_\mathsf{key}$*, public key* pk *and challenge ciphertext* C^* *are identically distributed in simulation and* $\mathsf{IND\text{-}CCA}$ *experiment. Furthermore,*

$$\Pr\left[\mathsf{bad}_\mathsf{key}\right] \leq \frac{2\ell_\mathsf{T}+3}{2^{k-1}}, \tag{22}$$

both in the simulation and in the $\mathsf{IND\text{-}CCA}$ *experiment.*

Thus, conditioned on $\neg\mathsf{bad}_\mathsf{key}$, D perfectly simulates A's input as in the real $\mathsf{IND\text{-}CCA}$ experiment if $V = \mathsf{BBS}_N(u) = \mathsf{BBS}_N(z^{\frac{1}{2^{\ell_\mathsf{K}}}})$, and as in the ideal $\mathsf{IND\text{-}CCA}$ experiment if V is random.

How to handle A's decryption queries. It remains to describe how D handles decryption queries of A as in the $\mathsf{IND\text{-}CCA}$ experiment. So say that A submits a ciphertext (R, S) for decryption. We may assume that $(R, S) \in \mathbb{Z}_N^* \times (\mathbb{Z}_N^* \cap [(N-1)/2])$. Let $t = (t_1, \ldots, t_{\ell_\mathsf{T}}) = \mathsf{T}(R)$. Write $d = |\{i : t_i \neq t_i^*\}| \in \{0, \ldots, \ell_\mathsf{T}\}$ for the Hamming distance of t and t^*, and abbreviate $\beta = \sum_{i=1}^{\ell_\mathsf{T}} \beta_{i,t_i}$. We call a ciphertext *consistent* iff the original decryption algorithm would not have rejected it. Hence, by (18), a ciphertext is consistent iff

$$S^2 = \left(R^2\right)^{\frac{\mathrm{dlog}_g \prod_{i=1}^k X_{i,t_i}}{2^{\ell_\mathsf{K}}}} \qquad \left(= \left(R^2\right)^{\frac{\beta 2^{\ell_\mathsf{K}} + \mathrm{dlog}_g h^d}{2^{\ell_\mathsf{K}}}} = \left(R^2\right)^{\beta + \frac{d}{2^{\ell_\mathsf{K}+1}L}}\right) \tag{23}$$

Our goal will be to implement a consistency check using our setup of variables above, and to decrypt consistent ciphertexts as in the $\mathsf{IND\text{-}CCA}$ experiment.

How to detect inconsistent ciphertexts. We first describe how D detects inconsistent ciphertexts. Namely, (23) is equivalent to

$$\left(S^2 (R^2)^{-\beta}\right)^{2^{\ell_\mathsf{K}+1}L} = (R^2)^d \tag{24}$$

since exponentiating with $2^{\ell_{\mathsf{K}}+1}L$ is a bijection on $\mathbb{QR}_N$,[5] and the group elements on both sides of (23) are squares. On the other hand, D can easily check (24) and hence efficiently detect and reject inconsistent ciphertexts.

How to decrypt consistent ciphertexts. Now assume that C is consistent and $t \neq t^*$ (so that $1 \leq d \leq \ell_{\mathsf{T}}$). Then,

$$\mathsf{BBS}_N\left(\left(S^2(R^2)^{-\beta}\right)^{\frac{L}{d}}\right) \overset{(24)}{=} \mathsf{BBS}_N\left(\left(R^2\right)^{\frac{1}{2^{\ell_{\mathsf{K}}+1}}}\right) = K. \tag{25}$$

Since $L/d \in \mathbb{N}$ by definition of $L = \operatorname{lcm}(1, \ldots, \ell_{\mathsf{T}})$, D can retrieve K efficiently using (25). Hence, D can decrypt all consistent ciphertexts satisfying $t \neq t^*$.

The case $t = t^*$. So let us turn to the case that $t = t^*$ and the ciphertext is consistent. Then, if $R = R^*$ holds, we have

$$S^2 \overset{(23)}{=} \left(R^2\right)^{\frac{\operatorname{dlog}_g \prod_{i=1}^{k} X_{i,t_i}}{2^{\ell_{\mathsf{K}}}}} \overset{(*)}{=} \left(R^{*2}\right)^{\frac{\operatorname{dlog}_g \prod_{i=1}^{k} X_{i,t_i^*}}{2^{\ell_{\mathsf{K}}}}} \overset{(20)}{=} S^{*2}, \tag{26}$$

where $(*)$ uses $R = R^*$ and $t = t^*$. Furthermore, $(R, S) \neq (R^*, S^*)$ implies $|S| = S \neq S^* = |S^*|$, so that $S \neq \pm S^*$ and $(S + S^*)(S - S^*) = S^2 - S^{*2} \overset{(26)}{=} 0 \bmod N$ yields a non-trivial factorization of N. Hence, D can efficiently factor N to solve its own input challenge (N, z, L) directly whenever $R = R^*$ and (R, S) is consistent.

On the other hand, if $\mathsf{T}(R) = t = t^* = \mathsf{T}(R^*)$ and $R \neq R^*$, then A has broken the target-collision resistance of T. Formally, let $\mathsf{bad}_{\mathsf{TCR}}$ denote the event that $t = t^*$ and $R \neq R^*$. If $\mathsf{bad}_{\mathsf{TCR}}$ occurs, D can safely give up, since

$$\mathsf{Pr}\left[\mathsf{bad}_{\mathsf{TCR}}\right] \leq \mathsf{Adv}^{\mathsf{TCR}}_{\mathsf{T},\mathsf{B}}(k) \tag{27}$$

for a suitable PPT adversary B on T that simulates D and A.

Summary of the decryption procedure. We summarize the decryption cases:

- inconsistent (R, S) (consistency check (24)$\Leftrightarrow$(23)$\Leftrightarrow$(18) not passed): reject,
- consistent (R, S) and $t \neq t^*$: decrypt using (25),
- consistent (R, S), $t = t^*$, and $R = R^*$: factor N (using $S \neq \pm S^*$ and $S^2 = S^{*2}$ by (26)),
- consistent (R, S), $t = t^*$, and $R \neq R^*$: give up simulation (A has found a T-collision).

Hence, also decryption is faithfully simulated unless $\mathsf{bad}_{\mathsf{TCR}}$ occurs.

Finishing the proof. We conclude that, unless $\mathsf{bad}_{\mathsf{TCR}}$ or $\mathsf{bad}_{\mathsf{key}}$ occurs, D perfectly simulates the real IND-CCA experiment upon input $V = \mathsf{BBS}_N(u)$,

[5] At this point we need that the modulus N consists of the product of two safe primes such that $\mathbb{QR}_N$ is a cyclic group whose order does not divide the integers $1, \ldots, \ell_{\mathsf{T}}$. Hence for KEM' it does not seem to be possible to avoid safe primes. (In contrast to KEM, cf. Section 5.)

and the ideal $\mathsf{IND\text{-}CCA}$ experiment if V is random. If we let D output whatever the simulated experiment outputs, we obtain:

$$\begin{aligned}\left|\Pr\left[\mathsf{D}(N, z, \mathsf{BBS}_N(u)) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{CCA\text{-}real}}_{\mathsf{KEM}',\mathsf{A}}(k) = 1\right]\right| &\le \Pr\left[\mathsf{bad}_{\mathsf{TCR}}\right] + \Pr\left[\mathsf{bad}_{\mathsf{key}}\right] \\ \left|\Pr\left[\mathsf{D}(N, z, U_{\{0,1\}^{\ell_{\mathsf{K}}}}) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{CCA\text{-}rand}}_{\mathsf{KEM}',\mathsf{A}}(k) = 1\right]\right| &\le \Pr\left[\mathsf{bad}_{\mathsf{TCR}}\right] + \Pr\left[\mathsf{bad}_{\mathsf{key}}\right].\end{aligned} \tag{28}$$

Using (22) and (27), Theorem 5 follows from (28).