A Variant of the Cramer-Shoup Cryptosystem for Groups of Unknown Order

Stefan Lucks

Theoretische Informatik, Universität Mannheim, 68131 Mannheim, Germany
lucks@th.informatik.uni-mannheim.de

Abstract. The Cramer-Shoup cryptosystem for groups of prime order is a practical public-key cryptosystem, provably secure in the standard model under standard assumptions. This paper extends the cryptosystem for groups of unknown order, namely the group of quadratic residues modulo a composed N. Two security results are: In the standard model, the scheme is provably secure if both the Decisional Diffie-Hellman assumption for QR_N *and* the factorisation assumption for N hold. In the random oracle model, the security of the scheme is provable by a quite efficient reduction.

1 Introduction

Security against chosen ciphertext attacks is essential for many cryptosystems. Naor and Yung [9] introduced this notion in the world of public-key cryptosystems and first described a scheme secure against non-adaptive chosen ciphertext ("lunchtime") attacks. Today, most cryptographers agree that a "good" public-key cryptosystem should be secure against *adaptive chosen ciphertext* (ACC) attacks. This notion has been introduced by Rackoff and Simon [10]. Dolev, Dwork and Naor [7] described a scheme provably secure against ACC attacks under standard assumptions. However, their scheme is too inefficient for practical applications. The research for provably secure and practically efficient cryptosystems has led to schemes provably secure in the *random oracle model* [2], and to schemes provably secure under non-standard assumptions such as the "oracle Diffie-Hellman" assumption [1].

The Cramer-Shoup cryptosystem [4] is the only cryptosystem known to be *both* practical *and* provably secure under standard assumptions – mainly, the decisional Diffie-Hellman assumption in groups of prime order. Recently, the same authors proposed a generalisation of their cryptosystem [5]. Its security can be based either on Paillier's decision composite residiosity assumption or on the (quite classical) quadratic residiosity (QR) assumption – or on the decisional Diffie-Hellman assumption in groups of prime order, as before. As pointed out in [5], the QR-based variant of the generalisation is not too efficient in practice.[1] In this paper, we deal with another variation, based on the Diffie-Hellman problem in specific groups of non-prime order.

Set $N = PQ$, $P = 2p + 1$, $Q = 2q + 1$, $p \neq q$, and let P, Q, p, and q be odd primes. In the remainder of this paper, we assume N to be of that form.

[1] A sample instantiation of the security parameters with $N \approx 2^{1024}$ in [5] implies the following: A public key needs 70 KB of storage space, and an encryption operation needs about 600 exponentiations modulo N. Note that this is for the QR-based variant only, the other variants are much more efficient.

Consider the group QR_N of the Quadratic Residues mod N and the Cramer-Shoup Cryptosystem in this group. ([4] originally proposed their cryptosystem for groups of prime order only.) As it will turn out, the legal user will not need to know the factorisation of N for either encryption, decryption or key generation (with the possible exception of generating an appropriate N itself). Since knowing the factorisation of N is equivalent to knowing the order of QR_N, the group QR_N may be *of unknown order* even for the legal user.

A security result in the standard model provides assurance against all attacks, while a random oracle security result only provides assurance against so-called "generic" attacks. On the other hand, it is desirable to base the security of cryptosystems on weak assumptions, instead of strong ones. In this spirit, Shoup [11] proposed a "hedged" variant of the Cramer-Shoup cryptosystem, being both provably secure in the standard model under a strong assumption *and* provably secure in the random oracle model under a weak assumption. In Section 7, we follow the same approach. Our extension is different from Shoup's technique, and the proof for the security in the random oracle model given here is more efficient than its counterpart in [11].

2 Properties of the Set $\mathbf{QR}_N$

In this section, we recall some number-theoretic terminology and facts. Let G be a finite multiplicative group of the *order* $|G| \geq 2$. The *order* $\mathrm{ord}(x)$ of $x \in G$ is the smallest integer $e > 0$ such that $x^e = x^0$. G is *cyclic*, if a *generator* g for G exists, i.e., an element $g \in G$ with $\mathrm{ord}(x) = |G|$. Further, $\{1\}$ and G itself are the two *trivial* subgroups of G, all other subgroups are nontrivial.

Recall that $N = PQ$, where $P = 2p+1$, $Q = 2q+1$, p and q are primes (i.e., both p and q are Sophie-Germain primes). Consider the set $\mathrm{QR}_N = \{x \in \mathbb{Z}_N^* \mid \exists a \in \mathbb{Z}_N^* : a^2 \equiv x \pmod N\}$ of Quadratic Residues modulo N. In the sequel, we use the following lemmas, which we prove in Section A of the appendix.

Lemma 1. *QR_N has a nontrivial subgroup of order p and a nontrivial subgroup of order q. Both subgroups are cyclic.*

Lemma 2. *QR_N is cyclic. It consists of one element of the order 1, $(p-1)$ elements of the order p, $(q-1)$ elements of the order q, and $(p-1)(q-1)$ elements of the order pq.*

Lemma 3. *For every $x \in QR_N$: $ord(x) \in \{p,q\} \Rightarrow gcd(x-1, N) \in \{P,Q\}$.*

Lemma 4. *Let g be a generator for QR_N. For every $x \in \mathbb{Z}_{pq}$: $ord(g^x) \in \{p,q\} \Leftrightarrow gcd(x,pq) \in \{p,q\}$.*

Computations in QR_N are computations modulo N. If it is implied by context, we omit writing explicitly "mod N" for multiplications or divisions mod N. If S is a finite set, we write $v \in_{\mathsf{R}} S$ if the value v is chosen from the set S according to the uniform probability distribution. We write $x \in_{\mathsf{R}} \mathbb{Z}_{pq}$ for choosing a random value x in $\mathbb{Z}_{pq}$ according a distribution statistically indistinguishable from uniform. We can, e.g., choose $x \in_{\mathsf{R}} \{0, \ldots, N^2\}$ and treat x like an element of $\mathbb{Z}_{pq}$.

3 Key Encapsulation Mechanisms

A key encapsulation mechanism (KEM) can be seen as the *secret-key part* of a hybrid cryptosystem. Combining a KEM with an appropriate secret-key cryptosystem provides the functionality of a public-key cryptosystem. If the secret-key cryptosystem satisfies some fairly standard security assumptions and the KEM is secure against ACC attacks, the public-key cryptosystem is secure against ACC attacks as well.[2] A KEM is a triple (Gen, KE, KD) of algorithms:

1. A *key pair generation* algorithm Gen, which, given a security parameter, randomly chooses a public-key/secret-key pair (PK,SK).
2. A randomised *key encapsulation* algorithm KE to choose (C, K)=KE(PK), i.e. a ciphertext C and an encapsulated key K.
3. A deterministic *key decapsulation* algorithm KD to compute K'=KD(SK,C), and to reject invalid ciphertexts.

A KEM is *sound*, if $K = K'$ for any (PK,SK)=Gen($\cdot$), (C, K)=KE(PK), and K'=KD(SK,C). The KEM presented below and its extension in Section 7 are both sound. Proving this is easy, but omitted here for the sake of space.

An *ACC attack* against a KEM (Gen, KE, KD) can be described by the following game:

1. A *key generation* oracle computes (PK,SK)=Gen($\cdot$) and publishes PK.
2. A *key encapsulation* oracle chooses (C, K)=KE(PK) and $\sigma \in_{\mathsf{R}} \{0, 1\}$. If $\sigma = 0$, the oracle sends (C, K) to the adversary, else (C, K') with $K' \in_{\mathsf{R}} \{0, 1\}^{|K|}$.
3. The adversary makes some queries $C_1, \ldots, C_q$ to a *key decapsulation* oracle, with $C_i \neq C$. For each query C_i, the oracle responds the value KD(SK,C_i), which may be either a bit string, or a special code to indicate rejection. For $i \in \{1, \ldots, q-1\}$, the adversary learns the response KD(SK,C_i), before she has to choose the next query C_{i+1}.
4. The adversary outputs a value $\sigma' \in \{0, 1\}$.

The *adversary's advantage* in guessing σ is the difference

$$|\mathrm{pr}[\sigma' = 1|\sigma = 1] - \mathrm{pr}[\sigma' = 1|\sigma = 0]|$$

of conditional probabilities. A KEM is *secure against ACC attacks*, or *ACC-secure* if, for all efficient adversaries, the advantage is negligible.

In Section B of the appendix, we compare ACC-secure KEMs with ACC-secure public-key cryptosystems and introduce lunchtime-security.

4 The Cryptosystem and some Assumptions

Here, we deal with the Cramer-Shoup cryptosystem and what assumptions we make to prove its security. Cramer and Shoup [4] considered groups G of (known) prime order q^*, while we consider the group QR_N of composed order pq. There is no need to actually know pq, not even for the owner of the secret key.[3] For the sake of simplicity, we restrict ourselves to describing the system as a key encapsulation mechanism, instead of a full scale public-key cryptosystem.

[2] This is called a "folk theorem" in [11]. See also [4, revised and extended version].

[3] Note that an adversary knowing pq can efficiently factorise N.

Cramer-Shoup cryptosystem in the group $\mathbf{QR}_N$:

- Key Generation Gen(l):
  - Generate N, P, Q, p, q as above with $2^{l-1} < N < 2^l$. Choose a generator g for QR_N.
  - Let $H : \{0,1\}^* \to \mathbb{Z}_m$ (with $m \leq pq$) be a secure hash function, as described below.
  - Randomly choose $w \in_{\mathsf{R}} \mathbb{Z}_{pq}$, and compute $g_2 = g^w$. Choose x_1, x_2, y_1, y_2, $z \in_{\mathsf{R}} \mathbb{Z}_{pq}$. Compute $c = g^{x_1} g_2^{x_2}$, $d = g^{y_1} g_2^{y_2}$, and $e = g^z$.
  - The public key is PK=(N, g, H, g_2, c, d, e). The secret key is SK=(x_1, x_2, y_1, y_2, z) in $\mathbb{Z}_{pq}^5$.
- Key Encapsulation KE(PK):
  - Choose $r \in_{\mathsf{R}} \mathbb{Z}_{pq}$, compute $u_1 = g^r$, $u_2 = g_2^r$, $k = e^r$, $\alpha = H(u_1, u_2)$ and $t = c^r d^{r\alpha}$.
  - The ciphertext is (u_1, u_2, t), the encapsulated key is k.
- Key Decapsulation[4] KD(SK,(U_1, U_2, T)) for $(U_1, U_2, T) \in \mathrm{QR}_N^2 \times \mathbb{Z}_N^*$:
  - Compute $K' = U_1^z$, $A' = H(U_1, U_2)$, $T' = U_1^{x_1 + y_1 A'} U_2^{x_2 + y_2 A'}$.
  - If $T = T'$ then output K', else reject.

Both in a group G of prime order and in composed order groups (such as QR_N and $\mathbb{Z}_N^*$), expressions such as $g^a * g^b$ and $(g^a)^b$ are equivalent to g^{a+b} and g^{ab}. For prime order groups "$a + b$" and "ab" are addition and multiplication in a *field*, but for general groups G these operations are defined in the *ring* $\mathbb{Z}_{|G|}$. Thus, the proof of security from [4] is not directly applicable to the cryptosystem proposed in the current paper, though our proof is along the same lines.

Assumption: (Target collision resistance of H)
Let F_H be a family of hash functions $\{0,1\}^* \to \mathbb{Z}_m$, for $m \leq pq$. Consider the following experiment:

- *Fix an input T for H (the "target").*
- *Randomly choose H from the family F_H.*

It is infeasible to find a "collision" for the target T, i.e., an input $T' \neq T$ such that $H(T) = H(T')$.
As a minor abuse of notation, we write "H is target collision resistant" ("TC-resistant") to indicate that H has has been chosen from such a family F_H.

Assumption: (decisional Diffie-Hellman (DDH) assumption for QR_N)
Recall that the generator g for QR_N is a public system parameter. Consider two distributions, the distribution R of triples $(g_2, u_1, u_2) \in_{\mathsf{R}} QR_N^3$ and the distribution D of triples (g_2, u_1, u_2) with $g_2 \in_{\mathsf{R}} QR_N$, $r \in_{\mathsf{R}} \mathbb{Z}_{pq}$, $u_1 = g^r$, and $u_2 = g_2^r$. It is infeasible to distinguish between R and D.

[4] We don't care if $T \notin \mathrm{QR}_N$, because $T' \in \mathrm{QR}_N$, and the test "$T = T'$" is supposed to fail if $T \notin \mathrm{QR}_N$. Section 8 describes how to enforce $U_1, U_2 \in \mathrm{QR}_N$.

Assumption: (computational Diffie-Hellman (CDH) assumption for QR_N)
Given two values $g_2 \in_{\mathsf{R}} QR_N$ and $u_1 \in_{\mathsf{R}} QR_N$ with $\log_g(u_1) = r$, it is infeasible to find the value $u_2 = g_2^r$.
Since $g \in \mathrm{QR}_N$ is a generator, $\log_g(x)$ is uniquely defined for $x \in \mathrm{QR}_N$.

Assumption: (factoring assumption for N)
Given N, it its infeasible to find P or Q.

Theorem 1 (Factoring $\Rightarrow$ CDH).
If the factoring N is infeasible, the CDH assumption for QR_N holds.

The proof is in Section C of the appendix.

5 Some Technicalities

Lemma 5. *Let g be a generator of QR_N and $w \in_{\mathsf{R}} \mathbb{Z}_{pq}$. The value $g_2 = g^w$ is a uniformly distributed random value in QR_N. With overwhelming probability, g_2 is a generator for QR_N.*

Proof. Clearly, g_2 is uniformly distributed. By Lemma 4, g_2 is a generator for $\mathrm{QR}_N \Leftrightarrow w \in \mathbb{Z}_{pq}^*$. Hence, pr[$g_2$ is a generator for QR_N] $= (p-1)(q-1)/pq$. □

Lemma 6. *If it is feasible to find any pair $(\alpha, \beta) \in \mathbb{Z}_{pq}$ with $(\alpha - \beta) \in \mathbb{Z}_{pq} - \mathbb{Z}_{pq}^* - \{0\}$, it is feasible to factorise N.*

Proof. Let g be a generator for QR_N. If $(\alpha - \beta) \in \mathbb{Z}_{pq} - \mathbb{Z}_{pq}^* - \{0\}$, then $\mathrm{ord}(g^{\alpha-\beta}) \in \{p, q\}$ and thus, we can compute $\gcd(g^{\alpha-\beta} - 1, N) \in \{P, Q\}$. □

Lemma 7. *Let g be a generator for QR_N and $g_2 \in_{\mathsf{R}} QR_N$. If it is feasible to choose u_1, u_2 such that $u_1 = g^{r_1}$, $u_2 = g_2^{r_2}$, and $(r_2 - r_1) \in \mathbb{Z}_{pq} - \mathbb{Z}_{pq}^* - \{0\}$, it is feasible to factorise N.*

Proof. Choose g_2 as suggested in Lemma 5: $w \in_{\mathsf{R}} \mathbb{Z}_{pq}$; $g_2 = g^w$. Since $(r_2 - r_1) \in \mathbb{Z}_{pq} - \mathbb{Z}_{pq}^* - \{0\}$, $\mathrm{ord}(g^{r_2-r_1}) \in \{p, q\}$. Similarly, $\mathrm{ord}(g_2^{r_2-r_1}) \in \{p, q\}$, and thus $\gcd(g_2^{r_2-r_1}, N) \in \{P, Q\}$. Due to $g_2^{r_2-r_1} = g_2^{r_2}/g_2^{r_1} = u_2/u_1^w$, and since we know w, we actually can compute $g_2^{r_2-r_1}$ and thus factorise N. □

Now we describe a simulator for the Cramer-Shoup cryptosystem. Its purpose is not to be actually used for key encapsulation and decapsulation, but as a technical tool for the proof of security. If an adversary mounts an attack against the Cramer-Shoup cryptosystem, the simulator may provide the responses, instead of an "honest" Cramer-Shoup oracle. Note that the adversary can make many key decapsulation queries, but only one single key encapsulation query.

A Simulator for the Cramer-Shoup Cryptosystem in QR_N:

- Generate the public key:
  - Let the values g, N and H and a triple $(g_2, u_1, u_2) \in \mathrm{QR}_N^3$ be given.

- Choose $x_1, x_2, y_1, y_2, z_1, z_2 \in_{\mathsf{R}} \mathbb{Z}_{pq}$. Compute $c = g^{x_1} g_2^{x_2}$, $d = g^{y_1} g_2^{y_2}$, and $e = g^{z_1} g_2^{z_2}$. [5] The public key is PK=(N, g, H, g_2, c, d, e).

– Key Encapsulation KE(PK):
  - Compute $k = u_1^{z_1} u_2^{z_2}$, $\alpha = H(u_1, u_2)$, $t = u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$.
  - The ciphertext is (u_1, u_2, t), the encapsulated key is k.

– Key Decapsulation KD(SK,(U_1, U_2, T)):
  - Compute $K' = U_1^{z_1} U_2^{z_2}$, $A' = H(U_1, U_2)$, $T' = U_1^{x_1+y_1 A'} U_2^{x_2+y_2 A'}$.
  - If $T = T'$ then output K', else reject.

6 A Proof of Security in the Standard Model

In this section, we prove the security of the Cramer-Shoup Cryptosystem in QR_N in the standard model. The proof is based on three lemmas.

Theorem 2 (Security in the standard model).
If H is TC-resistant and both the DDH assumption for QR_N and the factoring assumption for N hold, the Cramer-Shoup cryptosystem in QR_N is ACC-secure.

Lemma 8. *If the triple (g_2, u_1, u_2) given to the simulator is distributed according to distribution D, an adversary cannot statistically distinguish between the behaviour of the simulator and the Cramer-Shoup cryptosystem itself.*

Proof. If (g_2, u_1, u_2) is distributed according to D, a value r exists such that $u_1 = g^r$ and $u_2 = g_2^r$. We show that the simulator's responses are statistically indistinguishable from the *real* cryptosystem's responses.

Consider the key encapsulation query. The simulator computes

$$\begin{aligned}
&k = u_1^{z_1} u_2^{z_2} = g^{rz_1} g_2^{rz_2} = (g^{z_1} g_2^{z_2})^r = e^r, \\
&\alpha = H(u_1, u_2) \text{ and} \\
&t = u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = g^{rx_1+ry_1\alpha} g_2^{rx_2+ry_2\alpha} = g^{rx_1} g_2^{rx_2} g_2^{ry_1\alpha} g_2^{ry_2\alpha} = c^r d^{r\alpha}.
\end{aligned}$$

The distribution of the response $((g_2, u_1, u_2), k)$ is identical to the distribution of the cryptosystem's response.

Now consider the key decapsulation queries. If a query (U_1, U_2, T) is valid, i.e., if a value $R \in \mathbb{Z}_{pq}$ exists with $U_1 = g^R$ and $U_2 = g_2^R$, the simulator's response is the same as the response the cryptosystem provides. Both the simulator and the cryptosystem reject (U_1, U_2, T) if $T \neq T' = U_1^{x_1+y_1 A'} U_2^{x_2+y_2 A'}$, and else output $K' = u_1^R u_2^R = (g^R)^{z_1} (g_2^R)^{z_2} = (g^{z_1} g_2^{z_2})^R = e^R$. It remains to show that both the cryptosystem and the simulator (given (g_2, u_1, u_2) distributed according to D) reject all invalid key decapsulation queries with overwhelming probability – and thus essentially behave identically.

The decision to reject an invalid ciphertext (U_1, U_2, T) depends on four random values $x_1, x_2, y_1, y_2 \in \mathrm{QR}_N$. A part of the public key are the values c and d with $c = g^{x_1} g_2^{x_2} = g^{x_1} g^{wx_2}$ and $d = g^{y_1} g_2^{y_2} = g^{y_1} g^{wy_2}$, i.e.,

$$l_c := \log_g(c) = x_1 + wx_2 \iff x_1 = l_c - wx_2 \text{ and} \tag{1}$$

$$l_d := \log_g(d) = y_1 + wy_2. \iff y_1 = l_d - wy_2 \tag{2}$$

[5] In contrast to the simulator, the cryptosystem itself implicitly defines $z_2 = 0$.

These equations[6] provide public information about the quadruple (x_1, x_2, y_1, y_2) of secret values. The response to the encapsulation query provides another equation $\log_g(t) = rx_1 + ry_1\alpha + rwx_2 + rwy_2\alpha$, however $\log_g(t) = rl_c + rl_d\alpha$, i.e., this new equation linearly depends on Equations 1 and 2, and thus provides no new information about (x_1, x_2, y_1, y_2). This still leaves $(pq)^2$ possibilities for the quadruple (x_1, x_2, y_1, y_2).

Assume g_2 to be a generator for QR_N. (By Lemma 5, this is overwhelmingly probable.) Let the ciphertext (U_1, U_2, T) be invalid. Thus, $R_1 \neq R_2$ exist with $U_1 = g^{R_1}$ and $U_2 = g_2^{R_2}$. To answer the query, the values $K' = U_1^{z_1} U_2^{z_2}$ (or $K' = U_1^{z}$), $A' = H(U_1, U_2)$, and $T' = U_1^{x_1+y_1A'} U_2^{x_2+y_2A'} = g^{R_1x_1+R_1y_1A'} g_2^{R_2x_2+R_2y_2A'}$ are computed, which provides the equation

$$l_{T'} := \log_g(T') = R_1x_1 + R_1y_1A' + wR_2x_2 + wR_2y_2A'. \tag{3}$$

Equations 1 and 2 can be used to eliminate the variables x_1 and y_1:

$$\begin{aligned} l_{T'} &= R_1l_c - R_1wx_2 + R_1l_dA' - R_1wy_2A' + wR_2x_2 + wR_2y_2A' \\ &= R_1l_c + R_1l_dA' + wx_2(R_2 - R_1) + wy_2A'(R_2 - R_1) \end{aligned}$$

By Lemma 5 and Lemma 7 we know that with overwhelming probability and under the factoring assumption both w and $(R_2 - R_1)$ are invertible mod pq. If these two values are invertible, we may fix the value y_2 arbitrarily and there always exists a uniquely defined value

$$x_2 = \frac{l_{T'} - R_1l_c - R_1A'l_d - wy_2A'(R_2 - R_1)}{w(R_2 - R_1)}$$

to prevent the rejection of the invalid ciphertext (U_1, U_2, T). Each time an invalid ciphertext is rejected, this eliminates at most pq of the $(pq)^2$ possible quadruples (x_1, x_2, y_1, y_2). □

Lemma 9. *If the triple $(g_2, u_1, u_2) \in QR_N^3$ given to the simulator is distributed according to distribution R, the simulator rejects all invalid ciphertexts with overwhelming probability.*

Proof. Recall that the rejection of an invalid ciphertext (U_1, U_2, T) depends on the quadruple $(x_1, x_2, y_1, y_2) \in \mathrm{QR}_N$ of secret values, and that the public key provides the two linear Equations 1 and 2 to narrow down the number of possibilities for (x_1, x_2, y_1, y_2) to $(pq)^2$. The response to the encapsulation query provides the value $t = u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$ and thus a linear equation

$$l_t := \log_g(t) = r_1x_1 + r_1y_1\alpha + wr_2x_2 + wr_2y_2\alpha. \tag{4}$$

By using Equations 1 and 2, we can eliminate the variables x_1 and y_1:

$$\begin{aligned} l_t &= r_1l_c - r_1wx_2 + r_1l_d\alpha - r_1wx_2\alpha + wr_2x_2 + wr_2y_2\alpha \\ &= r_1l_c + r_1l_d\alpha + wx_2(r_2 - r_1) + wy_2\alpha(r_2 - r_1) \\ \Rightarrow y_2 &= \frac{l_t - r_1l_c - r_1l_d - wx_2(r_2 - r_1)}{w(r_2 - r_1)\alpha} \end{aligned}$$

[6] It is vital that l_c and l_d are uniquely defined. We need not actually compute l_c or l_d.

An invalid ciphertext (U_1, U_2, T) is rejected, except when Equation 3 holds, which means $T' = T$. Recall $\alpha = H(u_1, u_2)$ and $A' = H(U_1, U_2)$ and consider three cases:

- Case 1, $(U_1, U_2) = (u_1, u_2)$: By the definition of an ACC attack, we require $t \neq T$, and thus the key decapsulation query (U_1, U_2, T) will be rejected.
- Case 2, $(U_1, U_2) \neq (u_1, u_2)$ and $\alpha = A'$: This is a collision for H for the target (u_1, u_2), which contradicts the assumption for H to be TC-resistant.
- Case 3: $(U_1, U_2) \neq (u_1, u_2)$ and $\alpha \neq A'$: We have four unknowns $x_1, x_2, y_1, y_2 \in \mathrm{QR}_N$, and four Equations 1, 2, 3, and 4 describe their relationship. By solving this system of linear equations we get

$$y_2 = \frac{l_{T'} - r_1 l_c - \frac{l_t - r_2 l_c - r_1 l_d \alpha}{r_2 - r_1}(R_2 - R_1) - A' R_1 l_d}{(R_2 - R_1) w (A' - \alpha)},$$

which uniquely determines y_2 if all the four values $(r_2 - r_1)$, $(R_2 - R_1)$, w, and $(A' - \alpha)$ are invertible in $\mathbb{Z}_{pq}$.[7] The invertibility of $(r_2 - r_1)$ and $(R_2 - R_1)$ follows from Lemma 7, the invertibility of w follows from Lemma 5, and the invertibility of $(A' - \alpha)$ follows from Lemma 6. □

Lemma 10. *Let k be the encapsulated key in the response for the encapsulation query. If the triple $(g_2, u_1, u_2) \in QR_N^3$ given to the simulator is distributed according to distribution R, it is infeasible for the adversary to distinguish between k and a uniformly distributed random value.*

Proof. We set $r_1 = \log_g(u_1)$ and $r_2 = \log_{g_2}(u_2)$. Assume that $g_2 = g^w$ is a generator for QR_N and that $r_1 \neq r_2$. Both assumptions hold with overwhelming probability. Now we prove: *If all invalid decapsulation queries are rejected during the simulation, then under the factoring assumption it is infeasible for the adversary to distinguish between k and a random value.*

Observe that k only depends on the two random values $z_1, z_2 \in \mathrm{QR}_N$. Since $e = g^{z_1} g_2^{z_2}$, the public key provides one linear equation

$$l_e := \log_g(e) = z_1 + w z_2 \iff z_1 = l_e - w z_2. \tag{5}$$

The rejection of an invalid key decapsulation query does not depend on z_1 and z_2. If the decapsulation query (U_1, U_2, T) is valid and not rejected, we have a value R such that $U_1 = g^R$ and $U_2 = g_2^R$. By $\log_g(k) = Rz_1 + Rwz_2 = R \log_g(e)$ this provides another equation, linearly depending on Equation 5. The response for the key encapsulation query consists of a ciphertext (u_1, u_2, t) and a key $k = u_1^{z_1} u_2^{z_2} = g^{r_1 z_1} g^{w r_2 z_2}$, which provides a linear equation

$$l_k := \log_g(k) = r_1 z_1 + w r_2 z_2 = r_1 l_e - r_1 w z_2 + r_2 w z_2 = r_1 l_e + w z_2 (r_2 - r_1), \tag{6}$$

which finally gives

$$z_2 = \frac{l_k - r_1 l_e}{w(r_2 - r_1)}.$$

As before, we argue that with overwhelming probability and under the factoring assumption both w and $(r_2 - r_1)$ are invertible in $\mathbb{Z}_{pq}$. If w and $(r_2 - r_1)$ are invertible, then a unique value z_2 exists for every key $k \in \mathrm{QR}_N$. □

[7] This implies that the four linear equations 1, 2, 3, and 4 are linearly independent.

Proof (Theorem 2). If the adversary can break the cryptosystem by distinguishing a *real* encapsulated key from a *random* one, she can do so as well in the simulation, if the simulator input is chosen according to distribution D (Lemma 8). Since she cannot distinguish a real key from a random key in the simulation if the simulator input is distributed according to R (Lemmas 9 and 10), being able to break the cryptosystem means being able to distinguish distribution D from distribution R, contradicting the DDH assumption for QR_N. □

7 An Extension and its Security

We describe how to extend the Cramer-Shoup cryptosystem, dealing with a hash function h, which may be used like a random oracle (→ Figure 1):

Cramer-Shoup cryptosystem in $\mathbf{QR}_N$ with h-extension:

- The key pair (PK, SK) is the same as for the non-extended Cramer-Shoup cryptosystem. Let h be a function $h : \{1, 2, 3\} \times \mathrm{QR}_N^3 \to \mathrm{QR}_N$.
- Extend key encapsulation by computing
  $t_* = t * h(1, k, u_1, u_2)$ (→ solid arrows in Figure 1) and
  $\tau = t * h(2, k, u_1, u_2)$ and $k_* = k * h(3, \tau, u_1, u_2)$ (→ dashed arrows).
  The ciphertext is (u_1, u_2, t_*), the encapsulated key is k_*.
- Decapsulate the ciphertext $(U_1, U_2, T_*) \in \mathrm{QR}_N^2 \times \mathbb{Z}_N^*$ by computing K', T' as before, and reject if $T' * h(1, K', U_1, U_2) \neq T_*$. Else compute $\tau' = T' * h(2, K', U_1, U_2)$ and output $K'_* = K' * h(3, \tau', U_1, U_2)$.

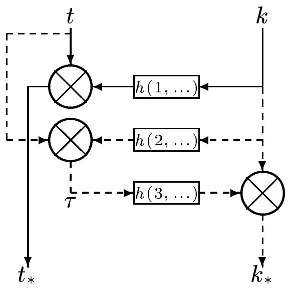

Fig. 1. The h-extension: converting t and h into t_* and h_*.

Theorem 3 (Security of h-extended scheme in standard model). *Let h be any efficient function $h : \{1, 2, 3\} \times QR_N^3 \to QR_N$. If H is TC-resistant and both the DDH assumption for QR_N and the factoring assumption for N hold, the Cramer-Shoup cryptosystem in QR_N is ACC-secure.*

Proof. Observe that the simulator described in Section 5 computes the values k and t when dealing with an encapsulation query. Also, being asked to decapsulate the ciphertext (U_1, U_2, T), the same simulator computes the values K' and T' from U_1 and U_2. Thus, it is straightforward to apply the h-extension to the simulator. Since h is efficient, the extended simulator is efficient, too.

Using the extended simulator instead of the original one, the proof for Theorem 2 is applicable to Theorem 3. □

Theorem 4 (Security of h-extended scheme in random oracle model). *If the function h is modelled as a random oracle, the h-extended scheme is ACC-secure under the factoring assumption.*

Proof. Let N and H be given. Consider an adversary with a non-negligible advantage to win the attack game. In the following experiment, we modify the key generation and we describe how to respond to the adversary's oracle queries, including queries to the random oracle. We start with the key generation:

- Choose $\beta \in_{\mathsf{R}} \{1, \dots, N^2\}$, $\alpha \in_{\mathsf{R}} \mathbb{Z}_N^*$ and compute $e := \alpha^2$.
- Choose $u_1 \in_{\mathsf{R}} \mathrm{QR}_N$ and compute $g := u_1^{2\beta}$.
  (We will be searching for $k = e^{\log_g(u_1)}$, i.e., for the value k with $k^{2\beta} = e$. If we find k, we have a 50 % chance that $\gcd(k^\beta - \alpha, N) \in \{P, Q\}$ holds, providing us with the factorisation of N.)
- Choose $w \in_{\mathsf{R}} \mathbb{Z}_{pq}$ and compute $g_2 = g^2$ and $u_2 = u^w$.
- Choose $x_1, x_2, y_1, y_2 \in_{\mathsf{R}} \mathbb{Z}_{pq}$ and compute $c = g^{x_1} g_2^{x_2}$ and $d = g^{y_1} g_2^{y_2}$.
- Use (N, g, H, g_2, c, d, e) as the public key.

The response to the key encapsulation query is the ciphertext (u_1, u_2, t_*) and the encapsulated key k_* with $t_*, k_* \in_{\mathsf{R}} \mathrm{QR}_N$.

Let (U_1, U_2, T_*) be a key decapsulation query. We respond as follows:

- Compute $T' = U_1^{x_1 + y_1 A'} U_2^{x_2 + y_2 A'}$.
- Consider values K' with queries for $\delta_1 = h(1, K', U_1, U_2)$ to the random oracle. Verify, if for one such value K' the equation

$$(K')^{2\beta} = U_1 \tag{7}$$

  holds. If not, or if $T_* \neq T' * \delta_1$, then reject.
- Else ask the random oracle for $\delta_2 = h(2, K', U_1, U_2)$, compute $\tau = T' * \delta_2$, ask for $\delta_3 = h(3, \tau, U_1, U_2)$, and respond $K'_* = K' * \delta_3$ to the adversary.

A random oracle query to compute $h(I, X, U_1, U_2)$ (with $I \in \{1, 2, 3\}$ and X, U_1, $U_2 \in \mathrm{QR}_N$) may be asked either by the adversary, or by ourselves when answering a key decapsulation query. The answer is computed as follows:

1. If we have been asked for $h(I, X, U_1, U_2)$ before, provide her with the same answer again.
2. Else, if $I \in \{1, 2\}$, $u_1 = U_1$, $u_2 = U_2$, and $X^{2\beta} = e$, print X and abort.
3. Else choose $Y \in_{\mathsf{R}} \mathrm{QR}_N$ and respond Y.

Observe that if we never abort ($\rightarrow$ Step 2), the adversary cannot distinguish h from a random function over the same domain. On the other hand, assume that we abort the experiment, having found a value X with $X^{2\beta} = e$, i.e., a square root (mod N) of e. Initially, we know two square roots of e, namely $\pm\alpha$. Since the adversary has no information about α, except for $e = \alpha^2$, $X^\beta \neq \pm\alpha$ holds with probability 1/2. In this case, we can factorise N by computing $\gcd(X^\beta - \alpha, N) \in \{P, Q\}$. This shows: *If π is the probability to abort the experiment, we can factorise N with the probability $\pi/2$ after running the experiment once.*

Now, we deal with three different games:

1. The attack game with the "real" encapsulated key k_*,
2. the attack game where k_* is replaced by a random value, and
3. the experiment we defined for the current proof.

As it turns out, the adversary cannot distinguish the experiment from either of the attack games, except when we abort the experiment:

- The public key values g, g_2, c, d, and e are independent uniformly distributed random values in QR_N – in the attack games, as in the experiment.
- In the attack games, the values u_1 and u_2 from the encapsulation query satisfy the equation $u_2 = g_2^{\log_g(u_1)}$, with $u_1 \in_{\mathsf{R}} \mathrm{QR}_N$. For one of the attack games, the values t_* and k_* depend on t and k (and h), while for the other one, t_* depends on t and k, while k_* is chosen at random.
  In the experiment, $u_1 \in_{\mathsf{R}} \mathrm{QR}_N$ and $u_2 = g_2^{\log_g(u_1)}$ as well. The value t_* cannot be distinguished from a uniformly distributed random value without asking for $h(1, k, u_1, u_2)$ (and then aborting). The value k_* cannot be distinguished without asking for $h(3, \tau, u_1, u_2)$. Asking this query is infeasible without having asked for $\delta_2 = h(2, k, u_1, u_2)$ (followed by an abortion), since τ depends on δ_2.
- Consider a decapsulation query (U_1, U_2, T_*). Let K' be defined by Equation 7. If h is a well-defined function, there is a unique well defined value T'_* such that a ciphertext (U_1, U_2, T'_*) has to be accepted, and every ciphertext (U_1, U_2, T_*) with $T_* \neq T'_*$ has to be rejected. Without asking for $h(1, K', U_1, U_2)$, the adversary cannot predict T'_*, and any ciphertext (U_1, U_2, T_*) chosen by the adversary is rejected with overwhelming probability in the attack games and with probability 1 in the experiment.
  If the adversary had asked for $h(1, K', U_1, U_2)$, the computation of T'_* and K'_* is exactly the same in the experiment as in the attack games. □

Efficiency of reduction: The proof of Theorem 4 provides an efficient reduction: *It turns any generic adversary for the cryptosystem with the advantage a into a factorising algorithm with the probability of success a/2.*

Comparison to Shoup's random oracle result [11]: When proving the security in the random oracle model, both [11] and the current paper describe a game to respond to the adversary's oracle queries, and both require to recognise oracle queries which correspond to a correct solution for a CDH problem in the given group. But [11] requires this recognition step to actually solve some instances of the DDH problem. This makes sense, since if the DDH problem is infeasible, the standard model security result applies. However, even if the DDH assumption turns out to be false, solving many instances of the DDH problem may be painfully slow in practice.[8] On the other hand, the recognition step for the current approach is quite efficient: Given X, compute $X^{2\beta}$ and compare the result with e. (See also Equation 7.) Thus, our reduction can be seen as more efficient and thus more meaningful than the one in [11].

[8] This is stressed in [4, revised and extended version].

Also note that we do not assume the hash function H to be TC-resistant, for Theorem 4, in contrast to [11, Theorem 3].

An advantage of the approach in [11] may be that it is based on the CDH assumption. If we generalise the technique from [11] for the group QR_N, we can replace the factoring assumption by the CDH assumption for QR_N, which is at least as strong as the factoring assumption for N, see Theorem 1.

Another difference to our approach is that [11] introduces the technical notion of a *pair-wise independent hash function* (PIH) and combines a PIH with a random oracle. The PIH is required for the security result in the standard model (i.e., for the counterpart of Theorem 3 in the current paper).

8 Final Remarks

The input for KD: Note that the input $(U_1, U_2, T^*) \in \mathrm{QR}_N^2 \times \mathbb{Z}_N^*$ is under control of the adversary. If x is a number, it is easy to verify whether $x \in \mathbb{Z}_N^*$, but it may be difficult to verify $x \in \mathrm{QR}_N$. We can deal with this problem by using KE' and KD' instead of KE and KD:

- KE': Compute KE and replace k by k^2 and t by t^2.
- KD'(SK,(U_1, U_2, T)) for $(U_1, U_2, T) \in (\mathbb{Z}_N^*)^3$: Compute KD(SK,$(U_1^2, U_2^2, T)$).

Note that (Gen,KE',KD') is as sound as (Gen,KE,KD). But for $(U_1, U_2, T) \in (\mathbb{Z}_N^*)^3$, the input for KD is now in $\mathrm{QR}_N^2 \times \mathbb{Z}_N^*$, as it should. A similar technique can be used for the h-extension.

The hash function H: Theoretically we don't need an *additional* assumption for the TC-resistance of H. Based on the factoring assumption, provably secure TC-resistant (and even stronger) hash functions are known. In practice, we may prefer to use a dedicated hash function such as SHA-1 or RIPE-MD 160.

Hiding the order of $\mathbf{QR}_N$: In this paper, we deal with a cryptosystem which's computations are in QR_N, but nobody knows (or needs to know) the order of QR_N. Our main reason for doing so is a technical one – we can proof the security of the cryptosystem using something like the factoring assumption. (Note that anyone knowing the order of QR_N can efficiently factorise N.) But there may actually be practical advantages of this approach, too. E.g., we may have different *insulated* key pairs (PK,SK) using the same modulus N. Information about the factorisation of N may serve as the secret *master key*. See [6] for introducing the notion of "key insulation".

Open problem: Kiltz [8] conjectured that Theorem 2 can be strengthened by avoiding the factoring assumption. Proving this is still an open problem.

Acknowledgement

The author is grateful to Eike Kiltz for useful discussions.

References

1. M. Abdalla, M. Bellare, P. Rogaway: "DHAES: an encryption scheme based on the Diffie-Hellman problem", preprint, `http://eprint.iacr.org`.
2. M. Bellare, P. Rogaway: "Random oracles are practical: a paradigm for designing efficient protocols", ACM Computer and Communication Security '93, ACM Press.
3. M. Bellare, A. Desai, D. Pointcheval, P. Rogaway: "Relations among notions of security for public-key encryption scheme", Crypto '98, Springer LNCS 1462.
4. R. Cramer V. Shoup: "A practical cryptosystem, provably secure against chosen ciphertext attacks", Crypto '98, Springer LNCS 1462. (Revised and extended version at `http://eprint.iacr.org`.)
5. R. Cramer V. Shoup: "Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption", preprint, 2nd version, December 12, 2001, `http://eprint.iacr.org`. (Extended abstract at Eurocrypt 2002.)
6. Y. Dodis, J. Katz, S. Xu, M. Yung: "Key-Insulated Public Key Cryptosystems", Eurocrypt 2002.
7. D. Dolev, C. Dwork, M. Naor: "Non-malleable cryptography", SIAM Journal of Computing, 2000. (Extended abstract at STOC '91, ACM Press.)
8. E. Kiltz, personal communication.
9. M. Naor, M, Yung: "Public-key cryptosystems provably secure against chosen ciphertext attacks", STOC '90, ACM Press.
10. C. Rackoff, D. Simon: "Non-interactive zero knowledge proof of knowledge and chosen ciphertext attacks", Crypto '91, Springer LNCS.
11. V. Shoup: "Using hash functions as a hedge against chosen ciphertext attack", Eurocrypt '00, Springer LNCS.

Appendix

A Properties of the Set QR_N – Proofs

In this section, we prove the Lemmas stated in Section 2. Consider the sets

$$\begin{aligned} \mathrm{QR}_P &= \{x \in \mathbb{Z}_P^* \mid \exists a \in \mathbb{Z}_P^* : a^2 \equiv x \pmod P\}, \\ \mathrm{QR}_Q &= \{x \in \mathbb{Z}_Q^* \mid \exists a \in \mathbb{Z}_Q^* : a^2 \equiv x \pmod Q\}, \text{ and} \\ \mathrm{QR}_N &= \{x \in \mathbb{Z}_N^* \mid \exists a \in \mathbb{Z}_N^* : a^2 \equiv x \pmod N\} \end{aligned}$$

of Quadratic Residues modulo P, Q and N. Recall the following facts (which we don't prove in the current paper):

Fact 1 *The sets QR_N, QR_P, and QR_Q are multiplicative groups.*

Fact 2 $|QR_N| = pq$, $|QR_P| = p$, *and* $|QR_Q| = q$.

Fact 3 *Groups of prime order are cyclic.*

Lemma 1. *QR_N has a nontrivial subgroup of order p and a nontrivial subgroup of order q. Both subgroups are cyclic.*

Proof. Note that $x \in \mathbb{Z}_N$ is in QR_N, if and only if x is both a Quadratic Residue mod P and a Quadratic Residue mod Q.

If $a \equiv 1 \pmod P$ and $b \equiv 1 \pmod Q$, then $ab \equiv 1 \pmod P$, and if both a and b are Quadratic Residues mod Q, then ab is a Quadratic Residue mod Q as well. Thus, the set

$$\{x \in \mathbb{Z}_N^* \mid \exists a \in \mathrm{QR}_Q : x \equiv a \bmod Q \text{ and } x \equiv 1 \bmod P\}$$

is a subgroup of QR_N of the order $|\mathrm{QR}_Q| = q$. Similarly, a subgroup of QR_N of the order p exists. Groups of prime order are cyclic. □

Lemma 2. *QR_N is cyclic. It consists of one element of the order 1, $(p-1)$ elements of the order p, $(q-1)$ elements of the order q, and $(p-1)(q-1)$ elements of the order pq.*

Proof. Consider $a, b \in \mathrm{QR}_N$ with $\mathrm{ord}(a) = p$, $\mathrm{ord}(b) = q$. Due to Lemma 1, such elements a and b exist; $\mathrm{ord}(ab) = \mathrm{lcm}(p, q) = pq$, thus $g = ab$ generates QR_N.

Due to $\mathrm{ord}(g) = pq$ we have $\mathrm{ord}(g^0) = \mathrm{ord}(g^{ab}) = \mathrm{ord}(1) = 1$, $\mathrm{ord}(g^i) = pq \Leftrightarrow (i > 1 \text{ and } \gcd(i, pq) = 1)$, $\mathrm{ord}(g^{kp}) = q$ for $k \in \{1, \ldots, q-1\}$, and $\mathrm{ord}(g^{lp}) = q$ for $l \in \{1, \ldots, p-1\}$. □

Lemma 3. *For every $x \in QR_N$: $ord(x) \in \{p, q\} \Rightarrow gcd(x-1, N) \in \{P, Q\}$.*

Proof. From Lemma 2 and the proof of Lemma 1: $\mathrm{ord}(x) = q \Leftrightarrow X \equiv 1 \pmod P \Rightarrow \gcd(x-1, N) = P$. Similarly: $\mathrm{ord}(x) = p \Rightarrow \gcd(x-1, N) = Q$. □

Lemma 3 implies that an adversary who is able to find any $x \in \mathrm{QR}_N$ with $\mathrm{ord}(x) \notin \{1, pq\}$, can factorise N. Further, if $\mathrm{ord}(x) = pq$, then $\gcd(x-1, N) = 1$. An implication of Lemma 2 is that it is easy to find a random generator for QR_N. Choose $x \in_{\mathsf{R}} \mathbb{Z}_N^*$ and compute $g = x^2 \bmod N$. If p and q are large, g is a generator for QR_N with overwhelming probability. In any case, g is a generator if and only if $\mathrm{ord}(g) \notin \{1, p, q\}$; $\mathrm{ord}(g) = 1 \Leftrightarrow g = 1$, and Lemma 3 provides a way to check for $\mathrm{ord}(g) \notin \{p, q\}$.

Lemma 4. *Let g be a generator for QR_N. For every $x \in \mathbb{Z}_{pq}$: $ord(g^x) \in \{p, q\} \Leftrightarrow gcd(x, pq) \in \{p, q\}$.*

Proof. If $x \equiv p \pmod{pq}$, then $g^{qx} = 1$ and thus $\mathrm{ord}(g^x) = q$. If $\mathrm{ord}(g^x) = q$, then $(g^x)^p = 1 \Rightarrow xp \equiv 0 \bmod pq \Rightarrow x \equiv p \pmod{pq}$. Thus, $x \equiv p \pmod{pq} \Leftrightarrow \mathrm{ord}(g^x) = q$. Similarly, we get $x \equiv q \pmod{pq} \Leftrightarrow \mathrm{ord}(g^x) = p$. □

B ACC-Secure Public-Key Cryptosystems and Lunchtime-Security

It may be interesting to point out the relationship to ACC-secure and lunchtime-secure public-key (PK) cryptosystems. Key decapsulation queries correspond to chosen ciphertext decryption queries in the PK world. The key encapsulation query corresponds to PK encryption query. Here, a plaintext is chosen by the adversary, the oracle either really encrypts that plaintext or it encrypts a random plaintext, and the adversary has to distinguish between real and random. Lunchtime (i.e. non-adaptive) security deals with *all* decryption queries *before* the encryption query. ACC attacks against PK cryptosystems deal with two phases of chosen ciphertext queries, the first *before* the encryption query, the second *after* the encryption query. [9] In the second phase, one may not ask for the decryption of the result of the encryption query.

A definition for a lunchtime-secure KEM would require a minor modification of our definition for an ACC-secure KEM by asking the decapsulation queries *before* the encapsulation query. And a two-phase attack against a KEM with some decapsulation queries before and some after the encapsulation query – similar to the ACC attack against PK cryptosystems – can easily be simulated by our (one-phase) ACC attack.

C The Proof for Factoring ⇒ CDH

Proof (Theorem 1). Assume the adversary with access to a CDH oracle for QR_N, i.e., given the modulus N and $g, g_2, u_2 \in_{\mathsf{R}} \mathrm{QR}_N$, the oracle computes $u_2 = g_2^{\log_g(u_1)}$ with probability π. Let N be given. We describe an algorithm to efficiently factorise N with probability $\approx \pi/2$, invocing the oracle once.

- Choose $\beta \in_{\mathsf{R}} \{1, \ldots, N^2\}$, $\alpha \in_{\mathsf{R}} \mathbb{Z}_N^*$ and compute $g_2 = \alpha^2$.
- Choose $u_1 \in_{\mathsf{R}} \mathrm{QR}_N$ and compute $g = u_1^{2\beta}$.
- Use the CDH oracle to compute u_2.
- If the ocacle succeeds, $u_2^{2\beta} \equiv g_2 \pmod{N}$.
- If $u_2^{\beta} \not\equiv \pm\alpha \pmod{N}$, print $\gcd(u_2^{\beta} - \alpha, N)$.

Since β is chosen from a set much larger than QR_N, the distribution of the values $g, g_2, u_2 \in \mathrm{QR}_N$ is statistically indistinguishable from uniform. Thus, the oracle finds u_2 with a probability of $\approx \pi$. Four square roots exist for $g_2 \in \mathrm{QR}_N$. Two such square roots are $\pm u_2^{\beta}$. The value α is a random square root of g_2. If $\alpha \not\equiv \pm g_2^{\beta} \pmod{N}$, then $\gcd(u_2^{\beta} - \alpha, N) \in \{P, Q\}$ factorises N. □

[9] Due to the number of phases, some authors denote lunchtime security by "IND-CCA1" and ACC security by "IND-CCA2" ("IND-" = "indistinguishable") [3]