An extension of the RSA trapdoor in a KEM/DEM framework

Bogdan Groza
Politehnica University of Timisoara
Faculty of Automatics and Computers
Bd. Vasile Parvan nr. 2, 300223 Timisoara, Romania
mail: bogdan.groza@aut.upt.ro

Abstract

A trapdoor based on an extension of the RSA trapdoor is proposed. The same function as in the RSA cryptosystem is used, i.e. \( x \mod n \), but there is no restriction for the exponent to be prime relatively to the order of the group while the function remains a permutation on a subgroup of \( \mathbb{Z}_n^* \). For the case when the exponent is not prime to the order of the group the resulting cryptosystem has its security equivalent to solving the integer factorization problem. This trapdoor is further used in a KEM/DEM (Key Encapsulation Mechanisms / Data Encryption Mechanisms) framework in order to obtain more efficient encryption and to achieve resistance against active adversaries. The resulting hybrid encryption scheme is provably secure against adaptive chosen ciphertext adversaries in the random oracle model.

1. Introduction

Public key cryptosystems are vital primitives for today’s information systems security. The basic task in which public key cryptosystems are involved, is in assuring the transfer of some secret information over a channel that is unsecured [7]. In order to achieve this objective a trapdoor one-way function can be used.

Although the concept of trapdoor one-way function is rather old, there are not very many candidates for this purpose [16], [15], [9], [13], [14], [17]. Therefore finding candidates for trapdoor one-way functions is itself a goal. In this paper we propose a trapdoor permutation based on what we call an extension of the RSA trapdoor. More concrete we use the same function as in the RSA cryptosystem, i.e. \( x^e \mod n \), but without requiring for the exponent to be prime relatively to the order of the group \( \mathbb{Z}_n^* \).

Another goal in building trapdoor permutations is establishing the number theoretic problem on which their security is based. For the case of the RSA cryptosystem still there is no proof on its equivalence to the integer factorization problem, and also there is some skepticism regarding the existence of such proof [5]. By using in our trapdoor exponents that are not relatively prime to the order of the group, equivalence to factoring can be proved. Also, since we perform encryption and decryption over some subgroup of \( \mathbb{Z}_n^* \) the function remains a trapdoor permutation on the elements of the subgroup. The fact that the trapdoor remains a permutation on the elements of a particular subgroup of \( \mathbb{Z}_n^* \) is an important feature since the Rabin trapdoor [15], which also uses exponents that are not prime to the order of the group, is not a permutation (except for the case when the modulus is a Blum integer).

Of course, today cryptosystems must achieve strong security goals as indistinguishability (IND) or non-malleability (NM) [8] against active adversaries such as adaptive chosen ciphertext adversaries (CCA2). IND assumes that an adversary cannot learn anything about the plaintext by having the corresponding ciphertext and NM assumes that an adversary cannot construct a valid ciphertext such that the corresponding plaintext is related in some known manner to the plaintext corresponding to some ciphertext given as challenge. In [2] these security notions for public key cryptosystems are formalized and relations between them established, in brief a scheme that is secure against IND-CCA2 adversaries is also secure against NM-CCA2 adversaries and vice versa.

In order to achieve these security objectives against active adversaries several generic conversion techniques were proposed, such as the RSA-OAEP [4] (which was initially assumed to be secure with any trapdoor, however a counter example was given to this with a xor-malleable function [18], while the use of the RSA trapdoor with OAEP remains secure [11] and [10]).

More recently, one-way trapdoors are used in KEM-DEM frameworks. Key Encapsulation Mechanisms (KEM) and Data Encryption Mechanisms (DEM) were proposed in [6], [12], [1] in order to build efficient and secure hybrid encryption schemes. Such a framework should have at least two merits: first it offers security against active adversaries...
and second encryption is more efficient since only a symmetric encryption key is encrypted with expensive asymmetric encryption, while the message itself is encrypted by a traditional symmetric primitive.

The paper is structured as follows. In section 2 the trapdoor that we are going to use is introduced and in section 3 its relation to the integer factorization problem is established. Section 4 uses this trapdoor in a KEM-DEM framework that is also used for the RSA function while in section 5 a formal proof of security is sketched. In section 6 we give a concrete practical example of such cryptosystem and section 7 holds our conclusion.

2. The trapdoor that we use

This section introduces the trapdoor permutation that we are going to use on the proposed cryptosystem. In brief, the trapdoor is constructed on a generalization of the discrete power function, i.e. \( f(x) = x^e \mod n \), that is used in the RSA and Rabin cryptosystems, for the case when the exponent is not necessarily prime to the order of the group. For the brevity of the exposition, we enumerate the following notations that will be used for the rest our paper:

- \( n = pq \) is the product of two large primes (an RSA like modulus)
- \( Z_n^* = \{ x | 0 < x < n, \gcd(x, n) = 1 \} \) (\( \gcd \) stands for greatest common divisor)
- \( \phi(n) = (p-1)(q-1) \) Euler phi function for integer \( n \) (also the order of the group \( Z_n^* \))
- \( \epsilon \) some integer exponent such that \( \gcd(\epsilon, \phi(n)) \geq 1 \)
- \( \lambda = \gcd(\epsilon, p-1), \mu = \gcd(\epsilon, q-1) \) the greatest common divisors between the exponent and the orders of the groups \( Z_p^* \) and \( Z_q^* \)
- \( \tau_{\text{min}} \) the minimal value of \( \tau \) for which it holds that \( \gcd(\phi(\tau), \phi(n), \epsilon) : \epsilon = 1 \)
- \( \delta'(n) = \frac{\phi(n)}{\gcd(\phi(n), \epsilon)} \) (note that \( \epsilon \) is relatively prime to \( \delta'(n) \))
- \( \delta' \) the multiplicative inverse of \( \epsilon \) in \( Z_n^* \), i.e. \( \delta' \epsilon \equiv 1 \mod \delta'(n) \)
- \( f(x) = x^e \mod n, f : Z_n^* \rightarrow Z_n^* \), \( g(x) = x^{\delta'} \mod n, g : Z_n^* \rightarrow Z_n^* \) (as will be shown next \( f \) is a trapdoor permutation on a particular subgroup of \( Z_n^* \) and \( g \) is its inverse)

Definition 2.1 Let \( Z_n^* \) denote the set of \( e^{th} \) residues modulo \( n \), i.e. \( Z_n^* = \{ x \in \mathbb{Z}_n^* | \exists y, x \equiv y^e \mod n \} \).

Theorem 2.2 If \( a \in Z_n^* \) then \( a^{\delta'} \mod n \) is one of its \( e^{th} \) roots, i.e. \( a \equiv (a^{\delta'})^{e \mod n} \).

Proof. According to the definition of \( Z_n^* \) if \( a \in Z_n^* \) there exists some integer \( b \) such that \( a \equiv b^{\delta e} \mod n \) and therefore \( a^{\delta e} \equiv b^{\delta e \mod n} \mod n \). But we have \( \delta' e \equiv 1 \mod \delta' \delta'(n) \mod n \) so that \( a^{\delta e \mod n} \equiv b^{\delta e \mod n} \mod n \). Now, since \( e' \equiv 0 \mod \phi(n) \) it leads to \( b^{\delta e \mod n} \equiv b^{\delta e \mod n} \equiv a^{\delta e \mod n} \) and this completes our proof.

Theorem 2.3 All elements from \( Z_n^* \) have exactly one \( e^{th} \) root that is also in \( Z_n^* \).

Proof. Let \( a \in Z_n^* \) then by using the result of theorem 2.2 we have that for \( b = a^{\delta'} \mod n \) it holds that \( a \equiv b^{\delta e} \mod n \). Since \( a \in Z_n^* \) there exists \( c \) such that \( a \equiv c^{\delta e} \mod n \) but \( b \neq d^{\delta e} \mod n \). Since \( b, d \in Z_n^* \) there must be some \( t \) such that \( d \equiv t^{e \mod n} \mod n \), \( c \equiv u^{e \mod n} \mod n \). Because \( b^{\delta e} \equiv d^{\delta e} \mod n \) we have \( t^{\delta e \mod n} \equiv u^{\delta e \mod n} \mod n \) and by raising at power \( \delta' \) we get \( t^{\delta e \mod n} \equiv u^{\delta e \mod n} \mod n \Rightarrow b \equiv d \mod n \) which completes our proof.

The following theorem introduces the trapdoor permutation that we are going to use.

Theorem 2.4 Function:

\[
f(x) = x^e \mod n, f : Z_n^{\tau_{\text{min}}} \rightarrow Z_n^{\tau_{\text{min}}}
\]

is a trapdoor permutation on \( Z_n^{\tau_{\text{min}}} \) and its inverse is:

\[
g(x) = x^{\delta'} \mod n, g : Z_n^{\tau_{\text{min}}} \rightarrow Z_n^{\tau_{\text{min}}}
\]

i.e. \( g(f(x)) = x \forall x \in Z_n^{\tau_{\text{min}}} \).

Proof. The result from 2.4 is a direct consequence of theorems 2.2 and 2.3 and no further proof is needed.

Remark 2.5 We note that if the exponent \( \epsilon \) is prime to the order of the group, i.e. \( \gcd(\epsilon, \phi(n)) = 1 \) (the case of a RSA exponent), then \( Z_n^{\tau_{\text{min}}} = Z_n^* \). Also we emphasize that for this case the RSA cryptosystem is more efficient than our proposal, however for this case the potential equivalence to the integer factorization problem can not be proved.

3. Relation to factoring

We want to establish the relation between the introduced trapdoor and the integer factorization problem in the case when the exponent is not prime to the order of the group, i.e. \( \gcd(\epsilon, \phi(n)) \neq 1 \). Of course, for the case when the exponent is prime to the order of the group (RSA cryptosystem) there is no proof of such equivalence, while for the case when \( \epsilon = 2 \) (the Rabin cryptosystem) it is commonly
known that computing square roots is equivalent to factoring. Therefore we are interested in the general case of our trapdoor, when the exponent is not necessarily 2, and it is some random integer such that \( \gcd(e, \phi(n)) > 1 \). More concrete, we want to prove that if there exists an algorithm for computing \( e^{th} \) roots in \( Z_n^* \) for such an exponent then this algorithm can be successfully used to factor \( n \).

First we recall the following two basic facts from number theory.

**Fact 3.1** Chinese Remainder Theorem (CRT): Let \( n_1, n_2, \ldots, n_k \) be integers that are pairwise relatively prime, then the system of congruences:

\[
\begin{align*}
x &\equiv a_1 \mod n_1 \\
x &\equiv a_2 \mod n_2 \\
& \vdots \\
x &\equiv a_k \mod n_k
\end{align*}
\]

has a unique solution modulo \( n = n_1 n_2 \ldots n_k \).

We do not give a proof for 3.1, since this is a commonly known fact in number theory. Also, it can be easily verified that the solution of the system can be efficiently computed as \( x = \sum_{i=1}^{k} N_i M_i x_i \mod n \), where \( N_i = n / n_i \) and \( M_i = N_i^{-1} \mod n_i \). Note that CRT describes an isomorphism between \( Z_n^* \) and \( Z_p^* \times Z_{q}^* \).

**Fact 3.2** If \( p \) is prime and \( d|p-1 \) then the equation \( x^d \equiv 1 \mod p \) has exactly \( d \) solutions.

This is also a basic aspect from number theory and proving is straightforward since \( x^{p-1} \equiv 1 \mod p \) has \( p-1 \) solutions in \( Z_p^* \) because by Fermat’s theorem this holds for any number from \( Z_p^* \). Now \( x^{p-1} \equiv 1 \Rightarrow x^{de} \equiv 1 \mod p \), if we assume that \( p-1 \equiv de \mod p \), and this leads to \( x^{de} \equiv 1 \Rightarrow (x^{d-1})^{e(e-2)+\ldots+x} \equiv 1 \mod p \).

But \( x^{d-1} \equiv 1 \mod p \) has at most \( d-1 \) solutions in \( Z_p^* \) (since it is a \( d-1 \) degree polynomial) therefore \( (x^{d-1})^{e(e-2)+\ldots+x} \equiv 1 \mod p \) and also has at most \( d \) solutions since it is a \( d \) degree polynomial - this means that \( x^{d-1} \) must have exactly \( d \) solutions.

The following theorem establishes the number of solutions for the equation \( x^e \equiv 1 \mod n \) when \( \gcd(e, \phi(n)) \neq 1 \) and its relation to the integer factorization problem.

**Theorem 3.3** The equation \( x^e \equiv 1 \mod n \) has exactly \( \lambda \mu \) distinct solutions and for \( \lambda + \mu - 2 \) of them it holds that \( \gcd(x_0 - 1, n) \) gives a non-trivial factor of \( n \); here \( x_0 \) denotes a solution of the equation, for other notations see section 2.

**Proof.** According to 3.2 the number of solutions to \( x^e \equiv 1 \mod n \) is \( \lambda \mu \) and the number of solutions to \( x^e \equiv 1 \mod n \) is \( \mu \) (note that we are in fact interested in the number of solutions to \( x^{\lambda \mu} \equiv 1 \mod n \) and \( x^{\lambda} \equiv 1 \mod n \) since \( x^\lambda \equiv 1 \mod n \) and \( x^\mu \equiv 1 \mod n \) are permutations and have only one solution). Now, due to the isomorphism of the Chinese Remainder Theorem each solution from \( Z_p^* \) can be grouped to each solution from \( Z_q^* \) giving exactly \( \lambda \mu \) distinct solutions in \( \mathbb{Z}_n^* \). Obviously if \( x_0 \equiv 1 \mod p \) and \( x_0 \equiv 1 \mod q \) or else \( x_0 \equiv 1 \mod p \) and \( x_0 \equiv 1 \mod q \) the value of \( \gcd(x_0 - 1, n) \) gives a non trivial factor of \( n \). It is straightforward to prove that it holds for exactly \( \mu - 1 \) solutions that \( x_0 \equiv 1 \mod p \) and \( x_0 \equiv 1 \mod q \) and for \( \lambda - 1 \) solutions that \( x_0 \equiv 1 \mod p \) and \( x_0 \equiv 1 \mod q \). This gives that a total of \( \lambda + \mu - 2 \) solutions that will give a non-trivial factor of \( n \) when computing \( \gcd(x_0 - 1, n) \).

**Theorem 3.4** If there exists an algorithm \( A \) for computing \( e^{th} \) roots in \( Z_{n}^{\tau_{min}} \) then there exist an algorithm \( A_{fact} \) that after one query to \( A \) gives a factor of \( n \) with probability \( Pr_{fact} = (1 - \frac{1}{\lambda \mu}) \frac{\lambda + \mu - 2}{n} \).

**Proof.** We assume that \( A \) behaves as follows: on input \( x \) it returns \( s \in Z_{n}^{\tau_{min}} \) such that \( x \equiv s^\mu \mod n \) and \( s \) does not exist. Then \( A_{fact} \) works as follows: it generates a new random integer \( r \in Z_n \), sets \( a_0 = r \), and computes \( a_1 = a_0^\mu \mod n \) and gives \( a_1 \) as input to \( A \). If \( A \) returns \( r \) (which is the same root that \( A_{fact} \) already knows) a new random \( r \) is generated and the same steps are followed - note that \( A \) returns \( r \) with probability \( \frac{1}{\lambda \mu} \) since all roots of \( a_1 \) are uniformly distributed in \( Z_n \). \( A \), \( A_{fact} \) either return a root \( v \) either \( \perp \). If \( A \) returns \( \perp \), then \( A_{fact} \) computes the array \( a_i = a_{i-1}^\mu \mod n, i > 1 \) and consecutively gives each \( a_2, a_3, \ldots, a_{\tau_{min}} \) as input to \( A \), until \( A \) does not return \( \perp \) anymore and returns \( s \) such that \( a_track \equiv s^\mu \mod n \) (obviously this will happen for some \( 0 < \tau < \tau_{min} \)). Since \( A \) returned \( \perp \) for \( a_{\tau-1} \) and \( v \in Z_{n}^{\tau_{min}} \) we have \( a_{\tau-1} \equiv v \) and \( a_{\tau-1}^\tau \equiv v^\tau \mod n \) from which one can compute \( (a_{\tau-1}^\tau)^e \equiv 1 \mod n \), \( (a_{\tau-1}^\tau)^e \equiv 1 \mod n \) as distinct roots of 1 in \( Z_n^{\tau_{min}} \). According to 3.3, the probability of the \( e^{th} \) root of 1 to give a non-trivial factor of \( n \) when computing \( \gcd(x_0 - 1, n) \) is \( \frac{\lambda + \mu - 2}{n} \) and therefore the probability of factoring with one query to \( A \), by \( A_{fact} \) is \( (1 - \frac{1}{\lambda \mu}) \frac{\lambda + \mu - 2}{n} \), repeated calls to \( A \) will increase this probability. Also, if the exponent is smaller, the probability of factoring is higher.

**Remark 3.5** Suppose that we are not in possession of an algorithm \( A \) that computes the \( e^{th} \) root of some element from \( Z^* \) and we have only an algorithm \( A_g \) that simply computes \( g(a) \) on any input \( a \in Z_n \). \( A_g \) does not return \( \perp \) if its input is not an element from \( Z^* \) in the case of \( A \), (basically \( A_g \) is the algorithm for computing \( g \) - the inverse of our trapdoor permutation). It is easy to prove that \( A_g \) may be used in the same manner as \( A \), to factor \( n \). Let \( a = r^e \mod n \) for some random integer \( r \) and \( b = g(a) \) the output of \( A_g \). It is trivial to prove that \( b^e \equiv r^e \) for some integer \( 0 \leq i \leq \tau_{min} \). If \( i = 0 \) the same root was returned and it cannot be used to factor the modulus (this happens again with probability \( \frac{1}{\lambda \mu} \)) otherwise, if \( i \neq 0 \), we obtain an \( e^{th} \) root of 1 which will lead to the factorization of the modulus with the same probability as previously \( \frac{\lambda + \mu - 2}{n} \).
4. The use in a KEM-DEM framework

Although the notion of hybrid encryption is known for many years, there was no formal treatment of the subject until [6]. The RSA-KEM proposed by Shoup in [19] is a straightforward approach that yields a secure KEM-DEM mechanism for the RSA trapdoor. The proposal is also close to the ideas from [3].

The same transformation can be used in the case of the trapdoor proposed previously. The following is the concrete mechanism for the RSA trapdoor. The proposal is also close to an approach that yields a secure KEM-DEM until [6]. The RSA-KEM proposed by Shoup in [19] is a

4. The use in a KEM-DEM framework

Although the notion of hybrid encryption is known for many years, there was no formal treatment of the subject until [6]. The RSA-KEM proposed by Shoup in [19] is a straightforward approach that yields a secure KEM-DEM mechanism for the RSA trapdoor. The proposal is also close to the ideas from [3].

The same transformation can be used in the case of the trapdoor proposed previously. The following is the concrete description of the KEM:

- **KEM.Gen(1^k):** Choose two distinct k bit primes p, q and an integer exponent e, compute n = pq, φ(n), τ_{min}, φ'(n), δ' (see section 2 for the significance of these values). Return pk = (n, e, τ_{min}) and sk = (n, δ).
- **KEM.Key(pk):** Choose at random an integer ω ∈ Z_n, compute γ = ω^{τ_{min}} mod n and the DEK key as dk = KDF(γ). Return (γ, dk).
- **KEM.Enc(γ):** Compute ψ = γ^e mod n. Return ψ.
- **KEM.Dec(ψ):** Compute γ = ψ^{σ'} mod n and kd = KDF(γ). Return dk.

Here KDF(x), KDF : Z_n → KD is some key derivation function, KD denotes the set where the DEK keys belong. In general, the security of KEM/DEM frameworks against active adversaries assumes that both components, KEM and DEM, are secure against active adversaries (it is a good intuition that if both components are secure then so is the KEM/DEM). If weaker assumptions need to be made on the DEM component, then a Tag-KEM/DEM framework can be used. The Tag-KEM is basically an enhancement of the KEM which consists in embedding a tag in the KEM in order to assure the non-malleability of the DEM. This proposal was also used in [17] and is also close to an approach from [3].

We can use the same mechanism with the proposed trapdoor. The only modifications that need to be done are on the KEM.Enc and KEM.Dec algorithms which need to take the tag into account. Thus for a Tag-KEM/DEM we have:

- **TagKEM.Enc(γ, T):** Compute ψ = γ^e mod n, ζ = H(γ, T). Return (γ, ζ).
- **TagKEM.Dec(ψ, ζ, T):** Compute γ = ψ^{σ'} and dk = KDF(γ). If H(γ, T) ≠ ζ return ⊥ else return dk.

In this construction for the tag T the symmetric encryption of some message will be used, i.e. T = Sym.Enc_{dk}(m), and H is some hash function.

Both the first KEM and the second Tag-KEM are close to the constructions E(x) = f(r)||G(r) ⊕ x and E(x) = f(r)||(G(r) ⊕ x)||H(rx) from [3], the later which is proved as non-malleable against chosen ciphertext adversaries in [3] (here G stands for a random generator, || denotes concatenation and ⊕ is logical XOR).

5. A formal proof of security

We give a sketch on the proof of security for the KEM that we used (detailed proofs on the security of such a KEM framework can be found in [19], also for the TagKEM/DEM proofs are in [1], while in [3] are also proofs for similar methods).

The CCA2 security of the KEM assumes that an adversary with adaptive access to a decryption oracle O has no chance in distinguishing that a given key is encapsulated or not in some challenge ψ = K.E.M.Enc(γ). Basically such an attack has four stages: first a public key, private key pair is generated, second the public key is given to the adversary and the adversary makes query to the oracle O, third a challenge is generated which consist in the encapsulation of a key and a key chosen at random between the correct key and some random key, fourth the adversary continues to make calls to the oracle (subject only to the restriction that it is not allowed to ask for the challenge itself) and then answers to the challenge.

The adaptive chosen ciphertext attack game can be easily formalized in the following steps:

1. (pk, sk) ← K.E.M.Gen(1^k)
2. v_1 ← As_k^G(pk)
3. (γ, dk_1) ← K.E.M.Key(pk), dk_0 ← KD, b ← \{0, 1\}, ψ ← K.E.M.Enc(γ)
4. b ← As_k^G(v_1, ψ, dk_0)

Here v_1 denotes state information of the adversary. The advantage of the attacker of the KEM can now be defined as ξ = |Pr[b = b] − 1/2|. The following theorem establishes the security bound for the KEM described in the previous section:

**Theorem 5.1** If there exists an adversary that can break the proposed KEM in the random oracle model with advantage ξ querying the decryption oracle q_D times then there exists an adversary that can factor integers with advantage ξ' ≥ (ξ - \frac{2}{n_p})(1 - \frac{1}{n_p}) and

**Proof.** The intuition is the following: we prove that an adversary that can break the proposed KEM can be used to compute \ell^b roots with advantage Adv = ξ - \frac{2}{n_p}, since each such root gives a probability of factoring equal to (1 - \frac{1}{n_p})^{\frac{λ + μ - 2}{λp}} the rest of the result is straightforward. Now
we want to prove that \( \text{Adv} \geq \xi - \frac{q_{22}}{n} \) in the random oracle model. Assuming that the hash function behaves as a random function, it is easy to simulate the decryption oracle to the adversary. This can be done as follows: a \( KL \) is prepared, on each input \( r \) the value of \( y = r^t \mod n \) is computed and a random value \( k \) is generated - these three values are stored in the \( KL \). If a ciphertext \( y \) is submitted to the decryption oracle, \( KL \) is inspected if the value of \( k \) is returned, otherwise a fresh key is generated and stored along the ciphertext, later if the decryption oracle is queried on some \( r \) such that \( y = r^t \mod n \) the value of \( k \) generated previously is returned. Suppose that in this environment an adversary gains infinitely many times an advantage \( \xi \). Let \( Awins \) denote the event that the adversary successfully guesses the hidden bit \( b \) and let \( Bad \) denote the event that the adversary queries the decryption oracle with the challenge ciphertext or the key derivation oracle with the value \( \gamma \). Now we have:

\[
\text{Pr}[Awins] = \text{Pr}[Awins \cap \text{Bad}] + \text{Pr}[Awins \cap \text{Bad}]
\]

Obviously, the oracle may be queried with the challenge only before the challenge is given to the adversary (because the adversary is restricted not to ask for the challenge), therefore this may happen with a probability bounded by \( \frac{q_{22}}{n} \). Let the event of asking for \( \gamma \) at \( KDF \) be denoted by \( \text{AskKDF} \), now it follows that:

\[
\frac{1}{2} + \xi \leq \text{Pr}[Awins \cap \text{Bad}] + \frac{q_{22}}{n} + \text{Pr}[\text{AskKDF}]
\]

If event \( \text{Bad} \) does not happen then all that the adversary knows is independent from the challenge, therefore:

\[
\text{Pr}[Awins \cap \text{Bad}] = \frac{1}{2}
\]

This finally leads to:

\[
\text{Pr}[\text{AskKDF}] \geq \xi - \frac{q_{22}}{n}
\]

The event \( \text{AskKDF} \) implies that the \( \epsilon \)th root was disclosed, and therefore we can use the adversary to compute \( \epsilon \)th roots with advantage \( \text{Adv} \geq \xi - \frac{q_{22}}{n} \). Since each such root gives a nontrivial factor with probability \((1 - \frac{1}{\lambda_\mu})(1 - \lambda_\mu^{-2})\) we have \( \xi' \geq (\xi - \frac{q_{22}}{n})(1 - \frac{1}{\lambda_\mu})(1 - \frac{1}{\lambda_\mu^{-2}}) \) and this completes our proof.

6. A practical example

The proposed cryptosystem is more efficient for the case when the value of \( \tau_{min} \) is smaller. Of course the most efficient case is when \( \tau_{min} = 0 \) which is the case of the RSA cryptosystem, unfortunately for this case equivalence to factorization cannot be proved. Therefore we are going to illustrate the case when \( \tau_{min} = 1 \) which is the most efficient case for which equivalence to factoring holds. Also we will use as encryption exponent \( \epsilon = 3 \) since computing \( x^3 \mod n \) will require only two modular multiplications. The description of the KEM is as follows:

- **\( \text{KEM.Gen}(1^n) \):** Choose two distinct \( k \) bit primes \( p, q \) such that \( p = 1 + 3k, k \not\equiv 0 \mod 3, q = 2 + 3l \) and set the integer exponent to \( \epsilon = 3 \). Compute \( n = pq \), \( \phi(n) = (p-1)(q-1) \), \( \tau_{min} \) which for the two primes previously defined will be \( \tau_{min} = 1 \), \( \phi'(n) = \frac{\phi(n)}{3} \), \( \delta' = 3^{-1} \mod \phi'(n) \). Return \( pk = (n, \epsilon, \tau_{min}) \) and \( sk = (n, \delta) \).

- **\( \text{KEM.Key}(pk) \):** Choose at random an integer \( \omega \in Z_n \), compute \( \gamma = \omega^3 \mod n \) and the DEM key as \( dk = KDF(\gamma) \). Return \( (\gamma, dk) \).

- **\( \text{KEM.Enc}(\gamma) \):** Compute \( \psi = \gamma^3 \mod n \). Return \( \psi \).

- **\( \text{KEM.Dec}(\psi) \):** Compute \( \gamma = \psi^{\delta'} \) and \( kd = KDF(\gamma) \). Return \( kd \).

A key encapsulated in this manner can be successfully used in a one-time scenario by any DEM that is secure against adaptive chosen ciphertext attacks. Also if there exists an IND-CCA2 attacker on this KEM then the attacker can compute \( 3^{th} \) roots in \( Z_n^* \) and each root distinct from one that is already known gives a non-trivial factor of \( n \) with probability \( \frac{\xi}{2} \) (this follows easily from theorem 3.3).

7. Conclusions

A trapdoor based on an extension of the RSA encryption function was constructed and its equivalence to integer factorization problem was established. The trapdoor works also for the cases when the exponent is not prime to the order of the group \( Z_n^* \) and for this case inverting the trapdoor gives a non-trivial factor of the modulus with high probability. The trapdoor was used in a KEM/DEM framework for achieving efficient hybrid encryption. The encryption is secure against active adversaries and a formal proof in the random oracle model was sketched. The main contribution of the paper is that the proposed trapdoor used in the KEM/DEM framework gives an efficient hybrid encryption scheme which has its security equivalent to the integer factorization problem.

**Acknowledgements:** This work was partially supported by national research grant MEDC-CNCSIS TD-122/2007.
References


