Lower Bounds on Black-Box Ring Extraction

Tibor Jager and Andy Rupp
tibor.jager@rub.de, arupp@crypto.rub.de

Horst Görtz Institute for IT Security
Ruhr-University Bochum, Germany

Abstract. The *black-box ring extraction problem* is the problem of extracting a secret ring element from a black-box by performing only the ring operations and testing for equality of elements. An efficient algorithm for the black-box ring extraction problem implies the equivalence of the discrete logarithm and the Diffie-Hellman problem. At the same time this implies the inexistence of secure ring-homomorphic encryption schemes.
It is unknown whether the known algorithms for the black-box ring extraction problem are optimal. In this paper we derive exponential-time lower complexity bounds for a large class of rings satisfying certain conditions. The existence of these bounds is surprising, having in mind that there are subexponential-time algorithms for certain rings which are very similar to the rings considered in this work. In addition, we introduce a novel technique to reduce the problem of factoring integers to the black-box ring extraction problem, extending previous work to a more general class of algorithms and obtaining a much tighter reduction.

1 Introduction

A famous question in cryptography is whether breaking the Diffie-Hellman protocol [1] is as hard as computing discrete logarithms. The discrete logarithm problem can be reduced to the Diffie-Hellman problem by assuming a *Diffie-Hellman oracle* DH solving the Diffie-Hellman problem, and showing that there exists an algorithm solving the discrete logarithm problem efficiently using DH. If the reduction algorithm is *generic*, i.e. does not exploit specific properties of a given representation of group elements, then such an algorithm would imply that breaking the Diffie-Hellman protocol is equivalent to computing discrete logarithms in *any* group.

Let $G = (\langle g \rangle, \circ)$ be a finite cyclic group of order n let σ be the isomorphism $\sigma : (\mathbb{Z}_n, +) \to G$ mapping $a \mapsto g^a$. In this notation the discrete logarithm problem in G is to compute $a \in \mathbb{Z}_n$ on input $\sigma(1), \sigma(a)$. Let us assume a Diffie-Hellman oracle DH for G that computes $\mathsf{DH}(\sigma(a), \sigma(b) = \sigma(ab)$. Then we may observe that the group G together with the Diffie-Hellman oracle is isomorphic to the *ring* $\mathbb{Z}_n$, i.e.,

$$(\langle g \rangle, \circ, \mathsf{DH}) \cong (\mathbb{Z}_n, +, \cdot)$$

where addition in $\mathbb{Z}_n$ can be performed by applying the group law

$$\sigma(a + b) = \sigma(a) \circ \sigma(b)$$

and the multiplication operation can be performed by querying the Diffie-Hellman oracle

$$\sigma(ab) = \mathsf{DH}(\sigma(a), \sigma(b)).$$

Thus we see that the question whether there exists a *generic* reduction from the discrete logarithm problem to the Diffie-Hellman problem corresponds to the question whether there exists an algorithm inverting any function σ by exploiting only the *structure* of the ring $(\langle g \rangle, \circ, \mathsf{DH})$.

Now let us consider an encryption function $\sigma : \mathsf{P} \to \mathsf{C}$ mapping elements of the plaintext space P to elements of the ciphertext space C. An encryption function σ is *homomorphic*, if

1. P and C exhibit an algebraic structure (P and C are groups, or rings, for instance),
2. the algebraic operations on P and C can be computed efficiently, and
3. σ is a bijective homomorphism from P to C.

There are several *group*-homomorphic encryption schemes, such as unpadded RSA [2], native ElGamal [3] or the Paillier scheme [4], for instance. However, it is a long-standing open question whether there exist *ring*-homomorphic encryption schemes, i.e. schemes which are homomorphic with respect to two operations such that the resulting algebraic structure forms a ring. An important step towards an answer of this question is to answer the following questions.

- Is the *structure* of a ring sufficient to invert any ring-homomorphic encryption function σ efficiently?
- Which conditions must be satisfied by a ring such that there is no efficient inversion algorithm?

Definition 1 (BBRE Problem). *The* black-box ring extraction problem *is the problem of inverting a ring-homomorphism σ by exploiting solely the structure of the given ring, without exploiting specific properties of a representation of ring elements.*

1.1 Related Work

Nechaev [5] and Shoup [6] studied the black-box extraction problem for *groups*, showing that any generic algorithm has to perform $\Omega(\sqrt{p})$ group operations to solve this problem, where p is the largest prime factor of the order of the group. This bound matches essentially the running time of well-known generic algorithms for the discrete logarithm problem [7, 8]. Hence the structure of a group is not sufficient to solve the black-box extraction problem efficiently (in general).

Boneh and Lipton [9] applied a technique due to Maurer [10] to describe an algorithm solving the black-box extraction problem over *prime fields* $\mathbb{F}_p$ in subexponential-time in $\log p$ under a (plausible) number-theoretic conjecture. Hence, in comparison to the results of Nechaev [5] and Shoup [6], the additional structure of a fields helps to solve the problem considerably more efficiently then in the case where the underlying algebraic structure is a group. It is unknown whether there are more efficient algorithms than the one proposed in [9]. Maurer and Raub [11] augmented the work of [9] from prime fields to *extension* fields, by reducing the BBRE problem over an extension field to the BBRE problem over the underlying prime field. As mentioned by Boneh and Lipton, the algorithm described in [9] can also be extended from prime fields to rings $\mathbb{Z}_n$ if n is a *squarefree* composite integer, by first factoring $n = \prod_{i=1}^{k} p_i$ (which can be done in subexponential time with current factoring algorithms) and solving the BBRE problem for all fields $\mathbb{Z}_{p_i}$, $i \in \{1, \ldots, k\}$. The solution for $\mathbb{Z}_n$ can then be obtained by Chinese remaindering.

Altmann et al. [12] have shown that there is no algorithm solving the BBRE for rings of characteristic n without (essentially) revealing a factor of n, thus in this sense the $\mathbb{Z}_n$-variant of the algorithm described in [9] is optimal. Hence, the black-box extraction problem can not be solved efficiently in general, unless there is an efficient algorithm for integer factorization. We note that Altmann et al. did not allow the BBRE-algorithm explicitly to compute multiplicative inverses. Since inverses can be computed efficiently in many rings (e.g. $\mathbb{Z}_n$), it seems to be an unfair restriction to exclude this operation.

Maurer and Wolf implicitly considered the BBRE problem for $\mathbb{Z}_n$ in [13], when showing that there is no efficient generic reduction from the discrete logarithm to the Diffie-Hellman problem in groups of order n, if n is divisible by a large multiple prime factor.

1.2 Our Contribution

Let $n \in \mathbb{N}$ and p be a prime. In this paper we derive exponential-time (in $\log p$) lower complexity bounds for the black-box extraction problem over rings where the characteristic n is divisible by p^2. To this end, we introduce a generalization of a technique used in [13]. The existence of these bounds appears quite surprising: Boneh and Lipton [9] show that for the ring $\mathbb{Z}_n$ there is an subexponential-time (in $\log n$) algorithm for the BBRE problem if n is squarefree. The result of [13] and this work imply that there is such an algorithm *only if* n is squarefree. In general, if $p^2 \mid n$ then any algorithm has to perform at least $O(\sqrt{p})$ ring operations to solve the BBRE problem. In contrast to [13], our result does not only cover the ring $\mathbb{Z}_n$, but any finite commutative unitary ring having characteristic n. This generalization is especially interesting with respect to the question of the existence of ring-homomorphic encryption schemes.

In addition, we introduce a novel technique to reduce the problem of factoring an integer n to the black-box ring extraction problem in the ring $\mathbb{Z}_n$. We obtain a much tighter reduction than previous work, and in addition allow for the computation of multiplicative inverses. In combination with [12], our result generalizes in a straightforward manner to arbitrary rings of characteristic n which are given in basis representation or as a polynomial ring. The reduction of integer factorization to the BBRE problem for $\mathbb{Z}_n$ proves a conjecture by Wolf [14, Conjecture 10.1], hence there is no efficient generic reduction from computing discrete logarithms to solving the Diffie-Hellman problem, unless factoring is easy.

2 The Black-Box Ring Extraction Problem

Let $(R, +, \cdot)$ be a finite ring, and let $B = \{r_1, \ldots, r_s\} \subseteq R$ be a set of generating elements of R, that is each element of R can be computed by performing a sequence of addition and multiplication operations on elements of B. We always assume that the ring R and elements of B are given in some explicit representation.

We adapt the generic group model of Shoup [6] to formalize the notion of black-box rings, and describe the BBRE problem in terms of a game between an algorithm $\mathcal{A}$ and a black-box ring oracle $\mathcal{O}$. The oracle encodes ring elements as randomly chosen bit strings from the set $S \subseteq \{0,1\}^{\lceil \log_2 |R| \rceil}$ with $|S| = |R|$, such that the mapping from R to S is a bijection. Hence each bit string is assigned uniquely to exactly one ring element. In order to maintain the assignment of encodings to ring elements, the oracle keeps two internal lists $\mathsf{L} \subseteq \mathbb{Z}_n$ and $\mathsf{E} \subseteq S$. In the following let L_i and E_i denote the i-th entry of L and E. E_i corresponds to the encoding of L_i. The oracle implements the internal procedures `compute`() and `encode`(), which are defined as follows.

- The `encode`()-procedure is called each time an element is appended to L. The procedure checks whether there exists a list element L_i, $i \in \{1, \ldots, |\mathsf{L}| - 1\}$, such that $L_{|\mathsf{L}|} \equiv L_i$. For the first i where this holds, the oracle sets $E_{|\mathsf{L}|} := E_i$. If there is no such i, a new encoding $E_{|\mathsf{L}|} \xleftarrow{\$} S \backslash \mathsf{E}$ is chosen at random from the set of unused encodings and appended to E.
- The `compute`()-procedure takes a triple $(i, j, \circ) \in \{1, \ldots, |\mathsf{L}|\} \times \{1, \ldots, |\mathsf{L}|\} \times \{+, -, \cdot\}$ as input. The procedure computes $\lambda = L_i \circ L_j$, appends λ to L, and calls the `encode`()-procedure.

Moreover, the oracle implements the public accessible functions `init`() and `query`(), which may be called by the algorithm $\mathcal{A}$. The public functions are defined as follows.

- When the `init`()-function is called, the lists L and E are set to the empty list. Then the oracle samples a uniformly random ring element $x \xleftarrow{\$} R$ and appends the elements of B and x to L, i.e. $L_i := r_i$ for $i \in \{1, \ldots, s\}$ and $L_{s+1} := x$, each time calling the `encode`()-procedure. The `init`()-function returns the triple $(E_1, \ldots, E_{s+1})$.

– The `query`()-function takes as input a triple $(E', E'', \circ) \in \mathsf{E} \times \mathsf{E} \times \{+, -, \cdot\}$. It determines the smallest $i, j \in \{1, \ldots, |\mathsf{L}|\}$ such that $E_i = E'$ and $E_j = E''$ and calls `compute`$(i, j, \circ)$. The query function returns the encoding $E_{|\mathsf{L}|}$ of the computed ring element.

At the beginning of the game the algorithm calls the `init`()-function and receives $(E_1, \ldots, E_{s+1})$ as input. Then it may call the `query`()-function to perform computations on encoded ring elements. We assume that the algorithm calls the `query`()-function at most m times. We say that $\mathcal{A}$ wins the game, if it outputs x in the given explicit representation of R.

3 Exponential Lower Bounds for BBRE

In the case that the characteristic of a ring is divisible by the square of a prime, we can prove a meaningful upper bound on the probability of any efficient black-box ring extraction algorithm:

Theorem 1. *Let R be a finite commutative unitary ring of characteristic* $\mathrm{char}(R) = \prod_{i=1}^{k} p_i^{e_i}$, *where $e_1 > 1$. Then any algorithm for the black-box ring extraction problem over R performing at most m oracle queries has a success probability of at most*

$$\frac{\lceil \log_2(|R|) \rceil (m + s + 1)^2 + 1}{p_1},$$

where $s = |B|$.

Hence, to achieve a certain success probability α, a BBRE algorithm needs to perform at least

$$\Omega\left(\sqrt{\frac{p_1 \alpha}{\log_2(|R|)}} - s\right)$$

operations.

To prove this theorem we make use of a technique introduced by Maurer and Wolf in [15]. Here, the authors consider the p-adic expansion of group elements to show that the DL problem over groups whose order is divisible by p^e (with $e > 1$) cannot be efficiently reduced to the DH problem. However, this technique cannot be applied immediately in our case but we have to choose an appropriate representation for ring elements first. More precisely, we decompose R into a certain product of rings and use the so-called basis representation for one of the component rings. These basics are introduced in Section 3.1. Then we apply the p-adic expansion technique to the elements in basis representation.

The general structure of our proof is typical of proofs in the black-box group or ring model: By means of a sequence of slight modifications, we transform the original black-box ring oracle $\mathcal{O}$ into an oracle $\mathcal{O}_{\mathsf{sim}}$ that simulates $\mathcal{O}$ without using the knowledge of a part of x. We call this setting the *simulation game*. Then we show that the behavior of $\mathcal{O}_{\mathsf{sim}}$ is perfectly indistinguishable from $\mathcal{O}$ unless a certain simulation failure $\mathbf{F}$ occurs. Denoting the success event of $\mathcal{A}$ when interacting with $\mathcal{O}$ and $\mathcal{O}_{\mathsf{sim}}$ by $\mathbf{S}$ and $\mathbf{S}_{\mathsf{sim}}$, respectively, we prove that $\Pr[\mathbf{S}]$ is upper bound by $\Pr[\mathbf{S}_{\mathsf{sim}}] + \Pr[\mathbf{F}]$. Section 3.2 describes the details of this part of the proof. Deriving concrete upper bounds on $\Pr[\mathbf{S}_{\mathsf{sim}}]$ and $\Pr[\mathbf{F}]$ in Section 3.3 completes the proof.

3.1 Some Preliminaries

In the context of the proof of Theorem 1 we make use of various representations of a finite ring which we are going to introduce in this section.

First, we need a convenient decomposition of finite commutative unitary rings which already proved its usefulness in [12]. Here, the authors showed that any such ring can be decomposed into a product of rings of prime power characteristic:

Theorem 2. *Let R be a finite commutative unitary ring of characteristic* $\mathrm{char}(R) = \prod_{i=1}^{k} p_i^{e_i}$. *Then R is isomorphic to a product of rings*

$$R_1 \times \cdots \times R_k$$

where R_i has characteristic $p_i^{e_i}$. We call such a decomposition of R a prime-power decomposition.

Furthermore, we consider the basis representation of a ring R. This is a tuple

$$(n_1, \ldots, n_t, (c_{i,j,\ell})_{1 \le i,j,\ell \le t}),$$

where $n_1, \ldots, n_t$ are the additive orders of elements $b_1, \ldots, b_t \in R$ generating $(R, +)$ and the $c_{i,j,\ell} \in \mathbb{Z}_{n_\ell}$ are integers describing the effect of multiplication on the b_i via

$$b_i b_j := \sum_{\ell=1}^{t} c_{i,j,\ell} b_\ell.$$

It is noteworthy that the integers n_i are factors of the characteristic of the ring R. The elements of R are represented by tuples

$$(r_1, \ldots, r_t) \in \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_t}.$$

The addition of two elements $r = (r_1, \ldots, r_t)$ and $u = (u_1, \ldots, u_t)$ in this representation is defined by component-wise addition, i.e.,

$$r + u = (r_1 + u_1, \ldots, r_t + u_t). \tag{1}$$

Multiplication is defined by

$$r \cdot u = \sum_{1 \le i,j \le t} (r_i u_j c_{i,j,1}, \ldots, r_i u_j c_{i,j,t}). \tag{2}$$

The two elements r, u are equal, which is denoted by $r \equiv u$, if and only if

$$r_i \equiv u_i \bmod n_i \text{ for all } 1 \le i \le t. \tag{3}$$

The elements $b_1 = (1, 0, \ldots, 0)$, $b_2 = (0, 1, 0, \ldots, 0)$, …, $b_t = (0, \ldots, 0, 1)$ represent an additive basis of R.

Remark 1. Note that the ring R is not necessarily isomorphic to the product ring $\mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_t}$ since multiplication is defined differently. However, if the basis representation satisfies

$$c_{i,j,k} = \begin{cases} 1, & i = j = k \\ 0, & \text{else} \end{cases}$$

then such an isomorphism exists. Indeed, the basis representation of R essentially corresponds to the canonical representation $\mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_t}$. As a special case, we obtain $R = \mathbb{Z}_n$ by setting $n_1 = n$ and $t = 1$.

3.2 Proof of Theorem 1: Introducing a Simulation Game

We first define an oracle $\mathcal{O}'$ which is equivalent to the original oracle $\mathcal{O}$ by doing some slight changes to $\mathcal{O}$. Then applying a minor modification to $\mathcal{O}'$ yields the simulation oracle $\mathcal{O}_{\mathsf{sim}}$.

An Equivalent Oracle

Modification 1. According to Theorem 2, the ring R can be decomposed as $R \cong R_1 \times \cdots \times R_k$ where $\mathrm{char}(R_i) = p_i^{e_i}$. Let $\psi : R \to R_1 \times \cdots \times R_k$ denote the corresponding isomorphism and $\psi_i : R \to R_i$ the surjective homomorphisms induced by ψ. Note that if r is a uniformly random element from R, then $\psi_i(r)$ is a uniformly random element from R_i. So our oracle $\mathcal{O}'$ can equivalently work with this product ring: The list $L \subset R_1 \times \cdots \times R_k$ is initialized with $\psi(r_1), \ldots, \psi(r_s), \psi(x)$. Multiplication, addition, subtraction in the `compute`()-procedure and equality checking in the `encode`()-procedure are performed component-wise.

Modification 2. Furthermore, we can use the following basis representation for R_1:

$$R_1 = (n_1, \ldots, n_t, (c_{i,j,\ell})_{1 \le i,j,\ell \le t}), \text{ where } n_1 = p_1^{e_1} \text{ and } n_{i+1} \mid n_i \,.$$

Thus, the n_i are all powers of p_1. The existence of such a representation follows from the fundamental theorem of finite abelian groups (e.g., see [16, Theorem 8.44]). Note that if a is a uniformly random element from R_1 also each component a_i of its basis representation is a uniformly random element from $\mathbb{Z}_{n_i}$. Arithmetic over R_1 works as defined by Equations (1) and (2). Equalities of elements over R_1 are checked according to Equation (3). In the following we use the simplifying notation $p := p_1$ and $e := e_1$.

Modification 3. Let us use the p-adic expansion to represent the components of the basis representation of elements from R_1. That means, given the basis representation $a = (a_1, \ldots, a_t) \in \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_t}$ of an element $a \in R_1$, we write each a_i as $a_i^{(0)}p^0 + \cdots + a_i^{(e-1)}p^{e-1}$, where $a_i^{(j)} \in \mathbb{Z}_p$. The values $a_i^{(j)}$ are called the digits of the p-adic expansion. Note that we have $a_i^{(j)} = 0$ for $p^{j+1} > n_i$. Moreover, the digits $a_i^{(j)}$ (where $p^{j+1} \le n_i$) of a uniformly random $a_i \in \mathbb{Z}_{n_i}$ are also uniformly random numbers from $\mathbb{Z}_p$.

Modification 4. Let $x \xleftarrow{\$} R$ be our secret ring element (given as input to $\mathcal{O}'$) and let

$$(x_1^{(0)}p^0 + \cdots + x_1^{(e-1)}p^{e-1}, \ldots, x_t^{(0)}p^0 + \cdots + x_t^{(e-1)}p^{e-1})$$

denote the representation of $\psi_1(x)$ as used internally by the oracle so far. We do another slight change to the representation of this element, namely we replace $x_1^{(e-1)}$ by some indeterminate X. That means, we initially insert

$$(x_1^{(0)}p^0 + \cdots + Xp^{e-1}, \ldots, x_t^{(0)}p^0 + \cdots + x_t^{(e-1)}p^{e-1})$$

as the representation of $\psi_1(x)$ into the list L. Arithmetic over R_1 is performed as before. Note that due to multiplications the indeterminate X may not only appear in the first component of the representation of an element from R_1 but may be part of possibly all components. For instance, consider the representation of $\psi_1(x^2)$:

$$\begin{aligned}
&\left((x_1^{(0)}p^0 + \cdots + Xp^{e-1})^2 c_{1,1,1}, \ldots, (x_1^{(0)}p^0 + \cdots + Xp^{e-1})^2 c_{1,1,t}\right) \\
&+ \sum_{2 \le j \le t} \left((x_1^{(0)}p^0 + \cdots + Xp^{e-1})(x_j^{(0)}p^0 + \cdots + x_j^{(e-1)}p^{e-1})c_{1,j,1}, \ldots, (x_1^{(0)}p^0 + \cdots + Xp^{e-1})(x_j^{(0)}p^0 + \cdots + x_j^{(e-1)}p^{e-1})c_{1,j,t}\right) \\
&+ \sum_{2 \le i \le t} \left((x_1^{(0)}p^0 + \cdots + Xp^{e-1})(x_i^{(0)}p^0 + \cdots + x_i^{(e-1)}p^{e-1})c_{i,1,1}, \ldots, (x_1^{(0)}p^0 + \cdots + Xp^{e-1})(x_i^{(0)}p^0 + \cdots + x_i^{(e-1)}p^{e-1})c_{i,1,t}\right) \\
&+ \sum_{2 \le i,j \le t} \left((x_i^{(0)}p^0 + \cdots + x_i^{(e-1)}p^{e-1})(x_j^{(0)}p^0 + \cdots + x_j^{(e-1)}p^{e-1})c_{i,j,1}, \ldots, (x_i^{(0)}p^0 + \cdots + x_i^{(e-1)}p^{e-1})(x_j^{(0)}p^0 + \cdots + x_j^{(e-1)}p^{e-1})c_{i,j,t}\right)
\end{aligned}$$

Obviously, it depends on the definition of the multiplication operation (more precisely, on the values $c_{i,j,\ell}$) in which components X can appear. Hence, $\mathcal{O}'$ internally uses tuples from $\mathbb{Z}_{n_1}[X]\times\cdots\times\mathbb{Z}_{n_t}[X]$ to represent elements from R_1.

To determine the encoding of a newly computed element $(a_1,\ldots,a_k)\in R_1\times\cdots\times R_k$ which should be appended to the list L, the following computation is triggered: The `encode`()-procedure of $\mathcal{O}'$ checks if there exists any index $1\le j\le|L|$ such that for $L_j=(L_{j,1},\ldots,L_{j,k})\in R_1\times\cdots\times R_k$ holds

$$((L_{j,1}-a_1)(x_1^{(e-1)}),L_{j,2}-a_2,\ldots,L_{j,k}-a_k)=(0,\ldots,0)\,.$$

If this equation holds for some j, then the respective encoding E_j is appended to E again. Otherwise, the `encode`()-procedure chooses a new encoding $s\xleftarrow{\$}S\setminus E$ and appends it to E. This completes the description of $\mathcal{O}'$. It is easy to see that $\mathcal{O}'$ and $\mathcal{O}$ are equivalent, i.e., their behaviors are perfectly indistinguishable. We say that an algorithm is successful when interacting with $\mathcal{O}'$ if it outputs x. This event is denoted by $\mathbf{S}$ in the following.

Modification 5. We slightly lighten the task of the algorithm $\mathcal{A}$: Let y denote the output of $\mathcal{A}$ and

$$\psi_1(y)=(y_1^{(0)}p^0+\cdots+y_1^{(e-1)}p^{e-1},\ldots,y_t^{(0)}p^0+\cdots+y_t^{(e-1)}p^{e-1})$$

be the representation of y's projection to R_1. We say that $\mathcal{A}$ is already successful if $y_1^{(e-1)}=x_1^{(e-1)}$. We denote this event by $\mathbf{S}'$. Clearly, we have $\Pr[\mathbf{S}]\le\Pr[\mathbf{S}']$.

The Simulation Oracle

Now we replace $\mathcal{O}'$ by a simulation oracle $\mathcal{O}_{\mathsf{sim}}$. The simulation oracle behaves exactly like $\mathcal{O}'$ except for the `encode`()-procedure, in order to be independent of $x_1^{(e-1)}$: Each time an element $(a_1,\ldots,a_k)$ should be appended the list L, $\mathcal{O}_{\mathsf{sim}}$ checks whether there exists an index $1\le j\le|L|$ such that for $L_j=(L_{j,1},\ldots,L_{j,k})\in R_1\times\cdots\times R_k$ holds

$$(L_{j,1}-a_1,L_{j,2}-a_2,\ldots,L_{j,k}-a_k)=(0,\ldots,0)\,. \tag{4}$$

Remember that the first component $(L_{j,1}-a_1)$ is a t-tuple of polynomials in the indeterminate X. If Equation (4) holds for some j, the encoding E_j is appended to E again.[1] If Equation (4) cannot be satisfied, the oracle $\mathcal{O}_{\mathsf{sim}}$ chooses a new encoding $s\xleftarrow{\$}S\setminus E$ and appends it to E.

In this way, the determined encoding and thus the behavior of $\mathcal{O}_{\mathsf{sim}}$ is independent of the part $x_1^{(e-1)}$ of the secret choice $x\xleftarrow{\$}R$ given as input to it. Obviously, the applied modification also leads to a behavior differing from that of $\mathcal{O}'$ in the case that there exist $(a_1,\ldots,a_k)$ and $(b_1,\ldots,b_k)$ in L such that

$$(a_1-b_1)(x_1^{(e-1)})\equiv 0\ \wedge\ a_1-b_1\not\equiv 0\,. \tag{5}$$

We denote this simulation failure event by $\mathbf{F}$. Note that if and only if $\mathbf{F}$ occurs, the simulation oracle computes different encodings for the same ring element. The success event $\mathbf{S}_{\mathsf{sim}}$ in the simulation game is defined like the event $\mathbf{S}'$.

It is easy to see that we have the following relationship between the considered events:

$$\mathbf{S}'\wedge\neg\mathbf{F}\iff\mathbf{S}_{\mathsf{sim}}\wedge\neg\mathbf{F}$$

[1] Observe that if this equation is satisfied for two different indices j and j' then we also have $E_j=E_{j'}$.

That means, both games proceed identically unless a simulation failure occurs. Using this relation we immediately obtain the desired upper bound on the success probability in the original game in terms of the success probability and the failure probability in the simulation game:

$$\Pr[\mathbf{S}] \leq \Pr[\mathbf{S}'] \leq \Pr[\mathbf{S}_{\mathsf{sim}}] + \Pr[\mathbf{F}]\,.$$

The last inequality follows from applying the Difference Lemma [17].

3.3 Proof of Theorem 1: Bounding Probabilities in the Simulation Game

Let us first bound the probability of the event $\mathbf{F}$. The crucial observation is that each of the tuples $(a_1 - b_1) \in \mathbb{Z}_{n_1}[X] \times \cdots \times \mathbb{Z}_{n_t}[X]$ considered in Equation (5) consists of polynomials of at most linear degree. This is due to the fact that a product of the form

$$(cXp^{e-1}) \cdot (c'Xp^{e-1})$$

vanishes modulo n_i for each i since n_i is a power of p that is smaller or equal to p^e. The probability that a non-zero linear polynomial in $\mathbb{Z}_{n_i}[X]$ evaluates to zero for uniformly random $x_1^{(e-1)} \in \mathbb{Z}_p$ is at most $\frac{1}{p}$. Hence, we get

$$\begin{aligned}
\Pr[\mathbf{F}] &= \Pr_{x_1^{(e-1)} \xleftarrow{\$} \mathbb{Z}_p}[\exists\ (a_1,\ldots,a_k),(b_1,\ldots,b_k) \in L :\ (a_1 - b_1)(x_1^{(e-1)}) \equiv 0\ \wedge\ (a_1 - b_1) \not\equiv 0] \\
&\leq \sum_{(a_1,\ldots,a_k),(b_1,\ldots,b_k)\in L} \Pr_{x_1^{(e-1)} \xleftarrow{\$} \mathbb{Z}_p}[(a_1 - b_1)(x_1^{(e-1)}) \equiv 0\ \wedge\ (a_1 - b_1) \not\equiv 0] \\
&\leq \sum_{(a_1,\ldots,a_k),(b_1,\ldots,b_k)\in L} \frac{t}{p} \\
&\leq (m+s+1)^2 \frac{t}{p}
\end{aligned}$$

Remember that t describes the number of components in which the group $(R_1, +)$ can be splitted according to the fundamental theorem of finite abelian groups. Clearly, t is upper bounded by $\lceil \log_2(|R_1|) \rceil$ which is in turn bounded by $\lceil \log_2(|R|) \rceil$.

Bounding the probability of $\mathbf{S}_{\mathsf{sim}}$ is simple: Since the computation in the simulation game is independent of the uniformly random $x_1^{(e-1)} \in \mathbb{Z}_p$, the algorithm virtually guesses the right output, i.e., we have

$$\Pr[\mathbf{S}_{\mathsf{sim}}] \leq \frac{1}{p}\,.$$

4 Reducing Integer Factorization to BBRE

Let us introduce a black-box ring oracle $\mathcal{O}_n$ for the ring $\mathbb{Z}_n$ that allows an explicit operation "/" to query multiplication with inverses. Oracle $\mathcal{O}_n$ is defined like the oracle $\mathcal{O}$ described in Section 2, but we adapt the description of the procedures to the ring $\mathbb{Z}_n$ and the additional operation "/". The oracle takes $x \xleftarrow{\$} \mathbb{Z}_n$ as input and implements the internal procedures `invertible`(), `compute`(), and `encode`(), which are defined as follows.

- The `invertible`()-procedure takes $L \in \mathsf{L}$ as input, and returns `true` if $L \in \mathbb{Z}_n^*$ and `false` if $L \notin \mathbb{Z}_n^*$..

– The `encode`()-procedure checks whether there exists a list element L_i, $i \in \{1, \ldots, |\mathsf{L}| - 1\}$, such that

$$L_{|\mathsf{L}|} \equiv L_i \bmod n.$$

– The `compute`()-procedure takes a triple $(i, j, \circ) \in \{1, \ldots, |\mathsf{L}|\} \times \{1, \ldots, |\mathsf{L}|\} \times \{+, -, \cdot, /\}$ as input. If $\circ = /$, then the the procedure `invertible`(L_j) is called; if `invertible`$(L_j) =$ `false` then the error symbol $\perp$ is returned. Otherwise the procedure computes $\lambda = L_i \circ L_j \bmod n$, appends λ to L, and calls the `encode`()-procedure.

The oracle implements the public accessible functions `init`() and `query`(), which are defined as follows.

– When the `init`()-function is called, the lists L and E are set to the empty list. Then the elements $1 \in \mathbb{Z}_n$ and x are appended to L, and the respective encodings (E_1, E_2) are returned.
– The `query`()-function takes as input a triple $(E', E'', \circ) \in \mathsf{E} \times \mathsf{E} \times \{+, -, \cdot, /\}$. It determines the smallest $i, j \in \{1, \ldots, |\mathsf{L}|\}$ such that $E_i = E'$ and $E_j = E''$ and calls `compute`$(i, j, \circ)$. If `compute`$(i, j, \circ) = \perp$, then the query function returns $\perp$. Otherwise the encoding $E_{|\mathsf{L}|}$ of the computed element is returned.

Let $n \in \mathbb{N}$ be the product of at least two different primes, and let $n = \prod_{i=1}^{k} p_i^{e_i}$ be the prime factor decomposition of n such that $\gcd(p_i, p_j) = 1$ for $i \neq j$.

Theorem 3. *Let $R = \mathbb{Z}_n$ with basis $B = \{1\}$. Suppose there exists an algorithm $\mathcal{A}$ solving the BBRE problem in R with probability ϵ by performing at most m operations in R. Then there exists an algorithm $\mathcal{B}$ finding a factor of n by running $\mathcal{A}$ and performing $O(m^2)$ operations in R and $O(m^2)$ gcd-computations on $\lceil \log_2 n \rceil$-bit numbers. Algorithm $\mathcal{B}$ has success probability at least $\epsilon - p^e/n$, where $e \geq 1$ and $p^e \mid n$ and $p^{e+1} \nmid n$.*

Proof. Suppose the black-box ring oracle $\mathcal{O}_n$ is replaced with an oracle $\mathcal{O}_{p^e}$. $\mathcal{O}_{p^e}$ is defined exactly like $\mathcal{O}_n$, except for the following modifications.

– The oracle takes a prime power p^e such that $p^e \mid n$ and $p^{e+1} \nmid n$ as additional input.
– The `invertible`()-procedure returns `true` if $L \in \mathbb{Z}_{p^e}^*$ and `false` if $L \notin \mathbb{Z}_{p^e}^*$.
– The `encode`()-procedure checks whether there exists a list element L_i, $i \in \{1, \ldots, |\mathsf{L}| - 1\}$, such that

$$L_{|\mathsf{L}|} \equiv L_i \bmod p^e.$$

– The `compute`()-procedure performs all computations modulo p^e.
– The `init`()-function appends the elements of B and $x \bmod p^e$ to L and returns $(E_1, \ldots, E_{s+1})$.

Note that all computations of $\mathcal{A}$ are performed in the subring $\mathbb{Z}_{p^e}$ of $\mathbb{Z}_n$. Thus the algorithm can not determine $x \in \mathbb{Z}_n$, but only $x \bmod p^e$. Hence the probability the success probability of any algorithm $\mathcal{A}$ when interacting with $\mathcal{O}_{p^e}$ is at most $\frac{1}{n/p^e} = p^e/n$. Also note that $\mathcal{O}_{p^e}$ simulates $\mathcal{O}_n$ perfectly, unless there exist $L_i, L_j \in \mathsf{L}$ such that $L_i \equiv L_j \bmod p^e$ and $L_i \not\equiv L_j \bmod n$. In this case a factor of n can be found by computing $\gcd(n, L_i - L_j)$.

It follows that the success probability ϵ of $\mathcal{A}$ in the original game is upper bounded by

$$\epsilon \leq p^e/n + \Pr[\mathbf{F}],$$

where $\mathbf{F}$ denotes the event that an interaction of $\mathcal{A}$ with $\mathcal{O}'$ is not indistinguishable from interaction of $\mathcal{A}$ with $\mathcal{O}$.

The factoring algorithm $\mathcal{B}$ implements oracle $\mathcal{O}$ (which does not receive a factor of n as input) and computes $\gcd(n, L_i - L_j)$ for all $i > j$ each time an element L_i is appended to L. Since there

are at most $(m+2)$ elements in L, this algorithm computes at most $(m+2)(m+1)/2 = O(m^2)$ differences $(L_i - L_j)$ and performs $O(m^2)$ gcd-computations on $\lceil \log_2 n \rceil$-bit numbers. Algorithm $\mathcal{B}$ is successful with probability at least

$$\epsilon - p^e/n.$$

□

From this theorem we may derive the following corollaries. The first one confirms a conjecture of Wolf [14, Conjecture 10.1], whereas the second one may be seen as evidence towards the existence of ring-homomorphic encryption schemes.

Corollary 1. *If there exists an efficient generic reduction from the discrete logarithm problem to the Diffie-Hellman problem, then there exists an efficient factoring algorithm.*

Remark 2. Note that the above corollary holds even if the reduction algorithm is given an additional oracle returning $\sigma(x^{-1})$ on input $\sigma(x)$. Such an oracle seems not to follow directly from a Diffie-Hellman oracle, unless $\phi(n)$ is known. At least in the $n = pq$ case, with p and q prime, computing $\phi(n)$ is as hard as factoring n.

Corollary 2. *If there exists an algorithm inverting any ring-homomorphic encryption function efficiently by only applying the ring operations, then there exists an efficient factoring algorithm.*

4.1 Generalizing Theorem 3

The proof of Theorem 3 can be generalized from the ring $\mathbb{Z}_n$ to finite commutative unitary rings of characteristic n that are given either in polynomial or in basis representation, by combining the technique presented in this work with results of Altmann et al. [12]. We sketch this in the following. Let $n = \prod_{i=1}^{k} p_i^{e_i}$ be the prime factor decomposition of n, and let R_n be a ring of characteristic n. R_n is decomposable into a direct product of rings $R_{p_1^{e_1}} \times \cdots \times R_{p_i^{e_k}}$ having prime power characteristic, cf. Theorem 2 . If the simulation oracle performs all computation in $R_{p_i^{e_i}}$ for some $i \in \{1, \ldots, k\}$, then a simulation failure occurs only if the algorithm has computed two ring elements $L_a, L_b \in R_n$ such that for $\Delta := (L_a - L_b)$ holds that $\Delta \neq 0 \in R_n$ and $\phi_{p_i^{e_i}}(\Delta) = 0 \in R_{p_i^{e_i}}$, where $\phi_{p_i^{e_i}} : R_n \to R_{p_i^{e_i}}$ is the surjective homomorphic projection mapping R_n to $R_{p_i^{e_i}}$. If R is given in basis or polynomial representation, then Altmann et al. [12] describe efficient algorithms that take as input Δ satisfying the above conditions and output a factor of n.

Acknowledgements. We thank the anonymous TCC 2009 program committee members for very good comments and positive feedback on our work.

References

1. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory **22** (1976) 644–654
2. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21** (1978) 120–126
3. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**(4) (1985) 469–472
4. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In Stern, J., ed.: EUROCRYPT. Volume 1592 of Lecture Notes in Computer Science., Springer (1999) 223–238
5. Nechaev, V.I.: Complexity of a determinate algorithm for the discrete logarithm. Mathematical Notes **55**(2) (1994) 165–172

6. Shoup, V.: Lower bounds for discrete logarithms and related problems. In Walter Fumy, ed.: Advances in Cryptology - EUROCRYPT 1997. Volume 1233 of Lecture Notes in Computer Science. (1997) 256–266
7. Shanks, D.: Class number, a theory of factorization, and genera. In Lewis, D.J., ed.: 1969 Number Theory Institute. Volume 20 of Proceedings of Symposia in Pure Mathematics., Providence, Rhode Island, American Mathematical Society (1971) 415–440
8. Pollard, J.M.: A Monte Carlo method for factorization. BIT **15** (1975) 331–334
9. Boneh, D., Lipton, R.J.: Algorithms for black-box fields and their application to cryptography (extended abstract). In Koblitz, N., ed.: CRYPTO. Volume 1109 of Lecture Notes in Computer Science., Springer (1996) 283–297
10. Maurer, U.M.: Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete algorithms. In Desmedt, Y., ed.: CRYPTO. Volume 839 of Lecture Notes in Computer Science., Springer (1994) 271–281
11. Maurer, U., Raub, D.: Black-box extension fields and the inexistence of field-homomorphic one-way permutations. In Kurosawa, K., ed.: ASIACRYPT. Volume 4833 of Lecture Notes in Computer Science., Springer-Verlag (2007) 427–443
12. Altmann, K., Jager, T., Rupp, A.: On black-box ring extraction and integer factorization. In Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I., eds.: ICALP (2). Volume 5126 of Lecture Notes in Computer Science., Springer (2008) 437–448
13. Maurer, U., Wolf, S.: The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. SIAM Journal on Computing **28**(5) (April 1999) 1689–1721
14. Wolf, S.: Information-theoretically and computationally secure key agreement in cryptography. PhD thesis, ETH Zurich (1999) ETH dissertation No. 13138.
15. Maurer, U.M., Wolf, S.: Lower bounds on generic algorithms in groups. In Nyberg, K., ed.: Advances in Cryptology - EUROCRYPT '98. Volume 1403 of Lecture Notes in Computer Science. (1998) 72–84
16. Shoup, V.: A Computational Introduction to Number Theory and Algebra. Cambridge University Press (2005)
17. Shoup, V.: Sequences of games: A tool for taming complexity in security proofs (2006) URL: `http://eprint.iacr.org/2004/332`.