Abstract Models of Computation in Cryptography

Ueli Maurer*

Department of Computer Science,
ETH Zurich, CH-8092 Zurich, Switzerland
maurer@inf.ethz.ch

Abstract. Computational security proofs in cryptography, without unproven intractability assumptions, exist today only if one restricts the computational model. For example, one can prove a lower bound on the complexity of computing discrete logarithms in a cyclic group if one considers only generic algorithms which can not exploit the properties of the representation of the group elements.

We propose an abstract model of computation which allows to capture such reasonable restrictions on the power of algorithms. The algorithm interacts with a black-box with hidden internal state variables which allows to perform a certain set of operations on the internal state variables, and which provides output only by allowing to check whether some state variables satisfy certain relations. For example, generic algorithms correspond to the special case where only the equality relation, and possibly also an abstract total order relation, can be tested.

We consider several instantiation of the model and different types of computational problems and prove a few known and new lower bounds for computational problems of interest in cryptography, for example that computing discrete logarithms is generically hard even if an oracle for the decisional Diffie-Hellman problem and/or other low degree relations were available.

1 Introduction and Motivation

1.1 Restricted Models of Computation

Proving the security of a certain cryptographic system means to prove a lower bound on the hardness of a certain computational problem. Unfortunately, for general models of computation no useful lower bound proofs are known, and it is therefore interesting to investigate reasonably restricted models of computation if one can prove relevant lower bounds for them.

In a restricted model one assumes that only certain types of operations are allowed. For example, in the monotone circuit model one assumes that the circuit performing the computation consists only of AND-gates and OR-gates, excluding NOT-gates. Such a restriction is uninteresting from a cryptographic viewpoint since it is obvious that an adversary can of course perform NOT-operations.

* Supported in part by the Swiss National Science Foundation.

N.P. Smart (Ed.): Cryptography and Coding 2005, LNCS 3796, pp. 1–12, 2005.

Nevertheless, some restricted models are indeed meaningful in cryptography, for example the generic model which assumes that the properties of the representation of the elements of the algebraic structure (e.g. a group) under consideration can not be exploited. In view of the fact that for some problems, for example the discrete logarithm problem on general elliptic curves, exploiting the representation is not known to be of any help and hence generic algorithms are the best known, such an assumption is reasonable from a practical viewpoint.[1]

The purpose of this paper is to provide a simple framework for such restricted models of computation and to prove some lower bounds. Generic algorithms are the simplest case. Some of the presented results are interpretations and generalizations of previous results, for instance of [10] and [4].

1.2 Generic Algorithms and Computing Discrete Logarithms

In order to compute with the elements of a set S (e.g. a group), one must represent the elements as bitstrings (without loss of generality). A representation is a bijective mapping from S to the set of bitstrings. A generic algorithm works independently of the representation. The term generic means that one can not exploit non-trivial properties of the representation of the elements, except for two generic properties that any representation has. First, one can test equality of elements, and second one can impose a total order relation $\preceq$ on any representation, for example the usual lexicographic order relation on the set of bitstrings. However, one can generally not assume that the representation is dense or satisfies any regularity or randomness condition.

In order to motivate the model to be introduced, we briefly discuss generic algorithms for computing discrete algorithms in a cyclic group G. A cyclic group G of order n, generated by a generator g, is isomorphic to the additive group $\mathbf{Z}_n$. A generic algorithm for computing the discrete logarithm (DL) x of an element $b = g^x$ to the base g in G can be modeled as follows. The algorithm is given a black-box which contains x. It can also input constants into the box[2] and add values in the box. The only information reported back from the box is when an equality (collision) between two computed elements occurs. The algorithm's task is to extract x by provoking collisions and computing x from the collision pattern. The order relation allows to establish ordered tables of the generated values and thus reduces the number of equality tests required, but it does not allow to reduce the number of computed values and is ignored in most of the following.

If one is interested in proving a lower bound on the number of operations for any generic algorithm, then one can consider the simpler objective of only provoking a *single* collision and that all equalities of elements are reported for free. Since only additions and the insertion of constants are allowed, every value computed in the box is of the form $ax + b$ (modulo n) for known values a and

[1] In contrast, for computing discrete logarithms in Z_p^* for a prime p, quite sophisticated algorithms are known (e.g. index calculus) which exploit that the elements are integers that can be factored into primes.

[2] One can also assume that the box contains only 1 and x initially and constants must be computed explicitly from 1 by an addition-and-doubling algorithm.

b. For uniform x the probability that two such values $ax + b$ and $a'x + b'$ collide is easily seen to be at most $1/q$, where q is the largest prime factor of n. Hence the total probability of provoking a collision is upper bounded by $\binom{k}{2}/q$ and therefore the running time of any algorithm with constant success probability is at least $O(\sqrt{q})$.

The simplest non-trivial generic DL algorithm is the so-called baby-step giant-step algorithm with complexity $O(\sqrt{n}\log n)$. It need not know the group order n, an upper bound on n suffices, and it is the best known algorithm when the group order is unknown. The Pohlig-Hellman algorithm [7] is also generic and a bit more sophisticated. It makes use of the prime factorization of n and has complexity $O(\sqrt{q}\log q)$, which is essentially optimal.

1.3 Discussion and Generalization of the Model

This view of a generic algorithm appears to be simpler than the model usually considered in the literature, introduced by Shoup [10], where one assumes that access to group elements is via a randomly selected representation. This complicates the random experiment in which the algorithm's success probability is to be analyzed. Also, in a realistic setting one has no guarantee that the representation corresponds in any way to a random mapping.

As a generalization of the described approach, one can also model that one can exploit certain additional information from the representation of the elements, for instance that one can test certain relations efficiently. As an example, one can imagine that one can efficiently test for any three elements x, y and z whether $xy = z$, which corresponds to assuming the availability of a decisional Diffie-Hellman (DDH) oracle. For this setting one can still prove an $O(\sqrt[3]{q})$ lower bound for the discrete logarithm problem.

2 An Abstract Model of Computation

2.1 The Model

We consider an abstract model of computation characterized by a black-box $\mathbf{B}$ which can store values from a certain set S (e.g. a group) in internal state variables $V_1, V_2, \ldots, V_m$. The storage capacity m can be finite or unbounded.

The initial state consists of the values of $V^d := [V_1, \ldots, V_d]$ (for some $d < m$, usually d is $1, 2$, or 3), which are set according to some probability distribution P_{V^d} (e.g. the uniform distribution).

The black-box $\mathbf{B}$ allows two types of operations, computation operations on internal state variables and queries about the internal state. No other interaction with $\mathbf{B}$ is possible.[3] We give a more formal description of these operations:

[3] This model captures two aspects of a restricted model of computation. The computation operations describe the types of computations the black-box can perform, and the state queries allow to model precisely how limited information about the representation of elements in S can be used. A quantum computer is another type of device where only partial information about the state can be obtained, but it could not be captured in our model.

- **Computation operations.** For a set Π of operations on S of some arities (nullary, unary, binary, or higher arity), a computation operations consist of selecting an operation $f \in \Pi$ (say t-ary) as well as the indices $i_1, \ldots, i_{t+1} \leq m$ of $t+1$ state variables.[4] $\mathbf{B}$ computes $f(V_{i_1}, \ldots, V_{i_t})$ and stores the result in $V_{i_{t+1}}$.[5]
- **Queries.** For a set Σ of relations (of some arities) on S, a query consist of selecting a relation $\rho \in \Sigma$ (say t-ary) as well as the indices $i_1, \ldots, i_t \leq m$ of t state variables. The query is replied by $\rho(V_{i_1}, \ldots, V_{i_t})$.

A black-box $\mathbf{B}$ is thus characterized by S, Π, Σ, m, and d. As mentioned above, one can include an abstract total order relation $\preceq$.

2.2 Three Types of Problems

We consider three types of problems for such black-boxes, where the problem instance is encoded into the initial state of the device.

- **Extraction:** Extract the initial value x of V_1 (where $d = 1$).[6]
- **Computation:** Compute a function $f : S^d \to S$ of the initial state within $\mathbf{B}$, i.e., one must achieve $V_i = f(x_1, \ldots, x_d)$ for some (known) i, where $x_1, \ldots, x_d$ are the initial values of the state variables $V_1, \ldots, V_d$.
- **Distinction:** Distinguish two black-boxes $\mathbf{B}$ and $\mathbf{B}'$ of the same type with different distributions of the initial state V^d.

An algorithm for solving one of these problems is typically assumed to be computationally unbounded, but it is restricted in terms of the number k of interactions with the black-box it can perform. The memory capacity m can also be seen as a parameter of the algorithm.

One is often only interested in the computation queries, especially when proving lower bounds, and can then assume that, for every (say t-ary) relation $\rho \in \Sigma$, $\mathbf{B}$ provides all lists $(i_1, \ldots, i_t)$ such that $\rho(u_{i_1}, \ldots, u_{i_t})$ for free. We prove lower bounds in this model.

The success probability of an algorithm is taken over the choice of the initial state $V_1, \ldots, V_d$ and the (possible) randomness of the algorithm. The advantage of a distinguisher is defined as usual.

3 Concrete Settings

In this section we consider a few concrete instantiations of the model which are of interest in cryptography.

[4] This information is the input to $\mathbf{B}$.

[5] A special case are constant functions, i.e., the operation of setting an internal state variable V_i to a particular value $c \in S$. If m is unbounded, then one can assume without loss of generality that each new result is stored in the next free state variable.

[6] More generally, one could consider the problem of extracting more general information about the initial state. This can be formalized by a function $g : S^d \to \mathcal{Q}$ for some $\mathcal{Q}$, where the task is to guess $g(V_1, \ldots, V_d)$.

3.1 Notation

We introduce some notation. Let $\mathcal{C}$ denote the set of constant (nullary) operations, which correspond to inserting a constant into the black-box. For a ring S, let $\mathcal{L}$ denote the set of linear functions (of the form $a_1V_1 + \cdots + a_dV_d$) on the initial state V^d. For a multiplicatively written operation (e.g. of a ring) S, let *square* denote the binary relation $\{(x, y) : y = x^2\}$, let *power*(e) denote $\{(x, y) : y = x^e\}$, and let *prod* denote the ternary relation $\{(x, y, z) : z = xy\}$.

For a given set Π of operations, let $\overline{\Pi}$ be the set of functions on the initial state that can be computed using operations in Π.

3.2 Extraction Problems with Constant and Unary Operations

The simplest case of an extraction problem to consider is when $\Pi = \mathcal{C}$ and $\Sigma = \{=\}$, i.e., one can only input constants and check equality.[7] It is trivial that the best strategy for the extraction problem is to randomly guess, and the success probability of any k-step algorithm is bounded by $k/|S|$, i.e., the complexity for achieving a constant success probability is $O(|S|)$. This bound holds independently of whether one counts equality checks or whether one assumes a total order $\preceq$ on S. This bound is trivially achievable with constant memory m.

If one would also allow to check a more general relation than equality (i.e., $\Sigma = \{=, \rho\}$ for some ρ), then better algorithms may exist. But the above upper bound generalizes easily to $kd/|S|$, where

$$d = \max_{u \in S} |\{v \in S : u \rho v \vee v \rho u\}|$$

is the maximal vertex degree of the relation graph. Note that $d = 1$ for the equality relation. If d is large, there can exist efficient algorithms. For example, if $\Sigma = \{=, \leq\}$ and S is totally ordered by the relation $\leq$, then one can use the binary search algorithm with running time $O(\log |S|)$, which is optimal.[8] It may be interesting to consider other relations.

We return to the case $\Sigma = \{=\}$ but now allow some unary operations.

Theorem 1. *Let $\star$ be a group operation on S, let $\Pi = \mathcal{C} \cup \{x \mapsto x \star a \mid a \in S\}$ consist of all constant functions and multiplications by constants, and let $\Sigma = \{=\}$. The success probability of every k-step algorithm for extraction is upper bounded by $\frac{1}{4}k^2/|S|$, and by $km/|S|$ if m is bounded.*

Proof. We use three simple general arguments which will be reused implicitly later. First, we assume that as soon as some collision occurs (more generally, some relation in Σ is satisfied for some state variables) in the black-box, the algorithm

[7] This corresponds to a card game where one has to find a particular card among n cards and the only allowed operation is to lift a card, one at a time.

[8] Note that the previously discussed order relation $\preceq$ can not be used to perform a binary search because it is not known explicitly, but only accessible through an oracle.

is successful.[9] One can therefore concentrate on algorithms for provoking some collision by computing an appropriate set of values in the black-box.

Second, we observe, as a consequence of Lemma 2 in Appendix B, that if the only goal is to provoke a deviation of a system from a fixed behavior (namely that it reports no collisions), then adaptive strategies are not more powerful than non-adaptive ones.

Third, for lower-bound proofs we can assume that an algorithm can not only perform operations in Π but can, in every step, compute a function in $\overline{\Pi}$ (of the initial state V^d). This can only improve the algorithm's power. Without loss of generality we can assume that only distinct functions are chosen by the algorithm.

In the setting under consideration, the composition of two operations in Π is again in Π, i.e., $\overline{\Pi} = \Pi$. For all $x \in S$ and distinct a and b we have $x \star a \neq x \star b$. Thus collisions can occur only between operations of the form $x \mapsto x \star a$ and constant operations. Let u and v be the corresponding number of operations the algorithm performs, respectively. Then the probability of a collision is upper bounded by $uv/|S|$. The optimal choice is $u = v \approx k/2$, which proves the first claim.

If m is finite, then in each of the k steps the number of potential collisions is at most $m-1$. The total number of x for which any of these collisions can occur is at most $k(m-1)$. △

The implied lower bound $k = O(\sqrt{n})$ for constant success probability can essentially be achieved even by only allowing a certain single unary operation, for example increments by 1 when $S = \mathbf{Z}_n$, i.e., $\Pi = \mathcal{C} \cup \{x \mapsto x + 1\}$. This is the abstraction of the baby-step giant-step (BSGS) algorithm: One inserts equidistant constants with gap $t \approx \sqrt{n}$ and increments the secret value x until a collision with one of these values occurs. If one considers a total order relation $\preceq$ one can generate a sorted table of stored values.[10]

3.3 The Group $\{0,1\}^\ell$

We consider the group $\{0,1\}^\ell$ with bit-wise XOR (denoted $\oplus$) as the group operation. As an application of Theorem 1 we have:

Corollary 1. *For $S = \{0,1\}^\ell$, $\Pi = \mathcal{C} \cup \{\oplus\}$ and $\Sigma = \{=\}$ the success probability of every k-step extraction algorithm is upper bounded by $\frac{1}{4}k^2 2^{-\ell}$.*

Proof. Any sequence of operations is equivalent wither to a constant function or the addition of a constant, i.e., the set $\overline{\Pi}$ of computable functions is $\overline{\Pi} = \mathcal{C} \cup \{x \oplus a \mid a \in \{0,1\}^\ell\}$. Hence we can apply Theorem 1. △

[9] Phrased pictorially, we assume a genie who provides x for free when any collision occurs.

[10] Note that the BSGS algorithm can also be stated as an algorithm for a group with group operation $\star$, where $\Pi = \{1, \star\}$, $\Sigma = \{=, \preceq\}$, and the addition operation is needed to compute other constants from the constant 1.

It is easy to give an algorithm essentially matching the lower bound of $O(2^{\ell/2})$ implied by the above corollary.

3.4 Discrete Logarithms in Cyclic Groups

We now consider the additive group $\mathbf{Z}_n$. The extraction problem corresponds to the discrete logarithm (DL) problem for a cyclic group of order n.[11]

In the sequel, let p and q denote the smallest and largest prime factor of n, respectively.

Theorem 2. *For $S = \mathbf{Z}_n$, $\Pi = \mathcal{C} \cup \{+\}$ and $\Sigma = \{=\}$ the success probability of every k-step extraction algorithm is upper bounded $\frac{1}{2}k^2/q$ and by km/q if the memory m is bounded.*

Proof. We have $\overline{\Pi} = \mathcal{L} = \{ax + b \mid a, b \in \mathbf{Z}_n\}$. As argued above, we need to consider only non-adaptive algorithms for provoking a collision. Consider a fixed algorithm computing in each step (say the ith) a new value $a_i x + b_i$, keeping $m-1$ of the previously generated values in the state. A collision occurs if $a_i x + b_i \equiv_n a_j x + b_j$ for some distinct i and j, i.e., if $(a_i - a_j)x + (b_i - b_j) \equiv_n 0$. Considered modulo q, this congruence has one solution for x (according to Lemma 1). The total number of x for which any collision modulo q (which is necessary for a collision modulo n) can occur is bounded by $k(m-1)$. If m is unbounded (actually $O(\sqrt{q})$ is sufficient), then the number of such x is bounded by $\binom{k}{2}$.[12] △

The case of unbounded m corresponds to the results of Nechaev [6] and Shoup [10], but the proof in [10] is somewhat more involved because a random permutation of the group representation is explicitly considered and makes the random experiment more complex. The Pohlig-Hellman algorithm requires $k = O(\sqrt{q}\log q)$ operations and essentially matches this bound. If the equality checks are also counted in k and no order relation is available, then $k = O(n)$ is required.

It is worthwhile to discuss the bounded-memory case. The theorem implies that the complexity of every algorithm achieving a constant success probability is $O(n/m)$, which is linear in n for constant m. Since memory is bounded in reality and $m = O(\sqrt{q})$ is typically infeasible, it appears that this result is a significant improvement of the lower bound over the unbounded memory case. However, this is in conflict with the fact that the Pollard-ρ algorithm [8] requires constant memory and also has (heuristic) complexity $O(\sqrt{q})$. The reason is that when a representation for S is explicitly available, then one can explicitly define a function on S, for example to partition the set S in a heuristically random manner into several subsets (three subsets in case of the Pollard-ρ algorithm). It is interesting to model this capability abstractly in the spirit of this paper.

[11] For other groups, such as $\{0,1\}^\ell$ discussed in the previous section, the extraction problem can be seen as a generalization of the DL problem.

[12] If no collision has occurred, one could allow the algorithm one more guess among the values still compatible with the observation of no collision, but this can be neglected.

3.5 The DL-Problem with a DDH-Oracle or Other Side Information

Let us consider the following natural question: Does a DDH-oracle help in computing discrete logarithms? Or, stated differently, can one show that even if the DDH-problem for a given group is easy, the DL-problem is still hard for generic algorithms. It turns out that the DDH oracle can indeed be potentially helpful, but not very much so.

Theorem 3. *For $S = \mathbf{Z}_n$, $\Pi = \mathcal{C} \cup \{+\}$ and $\Sigma = \{=, prod_n\}$ the success probability of every k-step extraction algorithm is upper bounded by $2k^3 + \frac{1}{2}k^2$. Every algorithm with constant success probability has complexity at least $O(\sqrt[3]{q})$.*

Proof. Each computed value is of the form $a_i x + b_i$ for some a_i and b_i. The product relation is satisfied for three computed values if

$$(a_i x + b_i)(a_j x + b_j) = a_k x + b_k$$

for some i, j, k, which is equivalent to

$$a_i a_j x^2 + (a_i b_j + a_j b_i - a_k)x + b_i b_j - b_k = 0,$$

a quadratic equation for x which has two solutions modulo q. There are k^3 such triples i, j, k. When also counting the potential collisions for the equality relation, the number of x modulo q for which one of the relations holds is bounded by $2k^3 + \binom{k}{2}$. △

A similar argument shows that when one considers a relation involving more than three variables, then the complexity lower bound decreases. For example, if we consider an oracle for the triple-product relation $\{(w, x, y, z) : z = wxy\}$, then we get a lower bound of $O(\sqrt[4]{q})$. It would be interesting to show that these bounds can be (or can not be) achieved.

A similar argument as those used above shows that when an oracle for the e-th power relation (i.e., $x_j = x_i^e$) is available, then every generic algorithm has complexity $O(\sqrt{q/e})$.

3.6 Product Computation in $\mathbf{Z}_n$ and the CDH Problem

We now consider the computation problem for the product function $(x, y) \mapsto xy$ in $\mathbf{Z}_n$. This corresponds to the generic computational Diffie-Hellman (CDH) problem in a cyclic group of order n analyzed already in [10]. Essentially the same bounds can be obtained for the squaring function $x \mapsto x^2$ in $\mathbf{Z}_n$. This theorem shows that for generic algorithms, the DL and the CDH problems are essentially equally hard.

Theorem 4. *For $S = \mathbf{Z}_n$, $\Pi = \mathcal{C} \cup \{+\}$ and $\Sigma = \{=\}$ the success probability of every k-step algorithm for computing the product function is upper bounded by $\frac{1}{2}(k^2 + 3k)/q$.*

Proof. Again, to be on the safe side, we can assume that as soon as a collision occurs among the values $a_i x + b_i$, the algorithm is successful. In addition, we need to consider the events $a_i x + b_i \equiv_n xy$ (for some i). For every i there are two solutions modulo q (according to Lemma 1). Hence the total number of x (modulo q) for which one of the collision events occurs is bounded by $\binom{k}{2} + 2k = \frac{1}{2}(k^2 + 3k)$. △

One can also show a $O(\sqrt[3]{n})$ generic lower bound for the CDH-problem when given a DDH-oracle.

3.7 Decision Problems for Cyclic Groups

We consider the decision problem for the squaring and product relations in $\mathbf{Z}_n$.

Theorem 5. *For $S = \mathbf{Z}_n$, $\Pi = \mathcal{C} \cup \{+\}$ and $\Sigma = \{=\}$ the advantage of every k-step algorithm for distinguishing a random pair (x, y) from a pair (x, x^2) is upper bounded by k^2/p.*

Proof. Again we can assume that as soon as a collision occurs among the values $a_i x + b_i$, the algorithm is declared successful. Hence it suffices to compute the probabilities, for the two settings, that a collision can be provoked, and take the larger value as an upper bound for the distinguishing advantage. For the pair (x, x^2) the set of computable functions is $\{ax^2 + bx + c \mid a, b, c \in \mathbf{Z}_n\}$, i.e., the ith computed value is of the form

$$a_i x^2 + b_i x + c_i$$

(in $\mathbf{Z}_n$) for some a_i, b_i, c_i. For any choice of $(a_i, b_i, c_i) \neq (a_j, b_j, c_j)$ we must bound the probability that

$$a_i x^2 + b_i x + c_i \equiv_n a_j x^2 + b_j x + c_j$$

for a uniformly random value x. This is equivalent to

$$(a_i - a_j)x^2 + (b_i - b_j)x + (c_i - c_j) \equiv_n 0.$$

There must be at least one prime factor p of n (possibly the smallest one) such that (a_i, b_i, c_i) and (a_j, b_j, c_j) are distinct modulo p. The number of solutions x of the equation modulo p is at most 2 (according to Lemma 1). Hence the total probability of provoking a collision modulo p (and hence also modulo n) is upper bounded by $\binom{k}{2} 2/p < k^2/p$.

This should be compared to the case where the pair (x, y) consists of two independent random values. The number of solutions (x, y) of

$$(a_i - a_j)y + (b_i - b_j)x + (c_i - c_j) \equiv_q 0$$

for any choice of $(a_i, b_i, c_i) \neq (a_j, b_j, c_j)$ is at most p. Hence the collision probability is, for all generic algorithms, upper bounded by $\binom{k}{2}/p < \frac{1}{2}k^2/p$. This concludes the proof. △

A very similar argument can be used to prove the same bound for the decision problem for the product relation, which corresponds to the generic decisional Diffie-Hellman (DDH) problem in a cyclic group of order n (see also [10]). To illustrate our approach we prove a lower bound for the DDH problem, even when assuming an oracle for the squaring relation.

Theorem 6. *For $S = \mathbf{Z}_n$, $\Pi = \mathcal{C} \cup \{+\}$ and $\Sigma = \{=, square_n\}$ the advantage of every k-step algorithm for distinguishing a random triple (x, y, z) from a triple (x, y, xy) is upper bounded by $\frac{5}{2}k^2/p$.*

Proof. We only analyze the case where the initial state is (x, y, xy). The set $\overline{\Pi}$ of computable functions is $\{ax + by + cxy + d \mid a, b, c, d \in \mathbf{Z}_n\}$, i.e., the ith computed value is of the form

$$a_i x + b_i y + c_i xy + d_i$$

for some a_i, b_i, c_i, d_i. For any choice of $(a_i, b_i, c_i, d_i) \neq (a_j, b_j, c_j, d_j)$ we must bound the probability that

$$a_i x + b_i y + c_i xy + d_i \equiv_n a_j x + b_j y + c_j xy + d_j$$

or that

$$(a_i x + b_i y + c_i xy + d_i)^2 \equiv_n a_j x + b_j y + c_j xy + d_j$$

The latter is a polynomial relation of degree 4 that is non-zero if $(a_i, b_i, c_i, d_i) \neq (a_j, b_j, c_j, d_j)$, except when $a_i = b_i = c_i = a_j = b_j = c_j = 0$ and $d_i^2 \equiv_n d_j$. However, we need not consider this case since it is known *a priori* that such a relation holds for all x and y.[13] The fraction of pairs (x, y) for which one of these relations can be satisfied modulo p is at most $5\binom{k}{2}/p$. △

3.8 Reducing the DL-Problem to the CDH-Problem

If one includes multiplication modulo n in the set Π of allowed operations for the generic extraction problem, i.e., one considers the extraction problem for the ring $\mathbf{Z}_n$, then this corresponds to the generic reduction of the discrete logarithm problem in a group of order n to the computational Diffie-Hellman problem for this group. The Diffie-Hellman oracle assumed to be available for the reduction implements multiplication modulo n. There exist an efficient generic algorithm for the extraction problem for the ring $\mathbf{Z}_n$ [3] (see also [5]) for most cases. For prime n the problem was called the black-box field problem in [1].

Acknowledgments

I would like to thank Dominic Raub for interesting discussions and helpful comments.

[13] More formally, this can be taken into account when defining the system output sequence to be deviated from according to Lemma 2.

References

1. D. Boneh and R. J. Lipton, Algorithms for black-box fields and their application to cryptography, *Advances in Cryptology - CRYPTO '96*, Lecture Notes in Computer Science, vol. 1109, pp. 283–297, Springer-Verlag, 1996.
2. W. Diffie and M. E. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.
3. U. Maurer, Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms, *Advances in Cryptology - CRYPTO '94*, Lecture Notes in Computer Science, vol. 839, pp. 271–281, Springer-Verlag, 1994.
4. U. Maurer and S. Wolf, Lower bounds on generic algorithms in groups, *Advances in Cryptology - EUROCRYPT 98*, Lecture Notes in Computer Science, vol. 1403, pp. 72–84, Springer-Verlag, 1998.
5. U. Maurer and S. Wolf, On the complexity of breaking the Diffie-Hellman protocol, *SIAM Journal on Computing*, vol. 28, pp. 1689–1721, 1999.
6. V. I. Nechaev, Complexity of a deterministic algorithm for the discrete logarithm, *Mathematical Notes*, vol. 55, no. 2, pp. 91–101, 1994.
7. S. C. Pohlig and M. E. Hellman, An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance, *IEEE Transactions on Information Theory*, vol. 24, no. 1, pp. 106–110, 1978.
8. J. M. Pollard, Monte Carlo methods for index computation mod p, *Mathematics of Computation*, vol. 32, pp 918–924, 1978.
9. J. T. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, *Journal of the ACM*, vol 27, no. 3, pp. 701–717, 1980.
10. V. Shoup, Lower bounds for discrete logarithms and related problems, *Advances in Cryptology - EUROCRYPT '97*, Lecture Notes in Computer Science, vol. 1233, pp. 256–266, Springer-Verlag, 1997.

A Polynomial Equations Modulo n

We make use of a lemma due to Schwartz [9] and Shoup [10] for which we give a simple proof.

Lemma 1. *The fraction of solutions $(x_1, \ldots, x_k) \in \mathbf{Z}_n$ of the multivariate polynomial equation $p(x_1, \ldots, x_k) \equiv_n 0$ of degree d is at most d/q, where q is the largest prime factor of n.*[14]

Proof. A solution of a multivariate polynomial equation $p(x_1, \ldots, x_k) \equiv_n 0$ over $\mathbf{Z}_n$ is satisfied only if it is satisfied modulo every prime factor of n, in particular modulo the largest prime q dividing n, i.e., $p(x_1, \ldots, x_k) \equiv_q 0$. It follows from the Chinese remainder theorem that the fraction of solutions $(x_1, \ldots, x_k)$ in $\mathbf{Z}_n^k$ is upper bounded by the fraction of solutions $(x_1, \ldots, x_k)$ in $\mathbf{Z}_q^k$.

Note that $\mathbf{Z}_q$ is a field. It is well-known that a univariate polynomial (i.e., $k = 1$) of degree $\leq d$ over a field F has at most d roots, unless it is the 0-polynomial for which all field elements are roots. The proof for multivariate

[14] The degree of a multivariate polynomial $p(x_1, \ldots, x_k)$ is the maximal degree of an additive term, where the degree of a term is the sum of the powers of the variables in the term.

polynomials is by induction on k. Let e be the maximal degree of x_k in any term in $p(x_1, \ldots, x_k)$. The polynomial $p(x_1, \ldots, x_k)$ over $\mathbf{Z}_n$ can be considered as a univariate polynomial in x_k of degree e with coefficients of degree at most $d - e$ in the ring $\mathbf{Z}_n[x_1, \ldots, x_{k-1}]$. By the induction hypothesis, for any of these coefficients the number of $(x_1, \ldots, x_{k-1})$ for which it is 0 is at most $(d-e)q^{k-2}$, which is hence also an upper bound on the number of tuples $(x_1, \ldots, x_{k-1})$ for which *all* coefficients are 0, in which case all values for x_k are admissible. If one of the coefficients is non-zero, then the fraction of solutions for x_k is at most e/q. Thus the total number of solutions $(x_1, \ldots, x_k)$ in $\mathbf{Z}_q$ is upper bounded by

$$(d-e)q^{k-2} \cdot q + (q-d+e)q^{k-2} \cdot e < dq^{k-1}. \qquad \triangle$$

B A Simple Lemma on Random Systems

Consider a general system which takes a sequence $X_1, X_2, \ldots$ of inputs from some input alphabet $\mathcal{X}$ and produces, for every input X_i, an output Y_i from some output alphabet $\mathcal{Y}$. The system may be probabilistic and it may have state.

Lemma 2. *Consider the task of provoking, by an appropriate choice of the inputs $X_1, \ldots, X_k$, that a particular output sequence $y^k := [y_1, \ldots, y_k]$ does* not *occur. The success probability of the best non-adaptive strategy (without access to $Y_1, Y_2, \ldots$) is the same as that of the best adaptive strategy (with access to $Y_1, Y_2, \ldots$).*

Proof. Any adaptive strategy with access to $Y_1, Y_2, \ldots$ can be converted into an equally good non-adaptive strategy by feeding it, instead of $Y_1, Y_2, \ldots$, the (fixed) values $y_1, \ldots, y_k$. As long as the algorithm is not successful, these constant inputs $y_1, y_2, \ldots$ correspond to what happens in the adaptive case. $\triangle$