Generic Lower Bounds for Root Extraction and Signature Schemes in General Groups[⋆]

Ivan Damgård and Maciej Koprowski[⋆⋆]

BRICS[⋆ ⋆ ⋆], Aarhus University

Abstract. We study the problem of root extraction in finite Abelian groups, where the group order is unknown. This is a natural generalization of the problem of decrypting RSA ciphertexts. We study the complexity of this problem for generic algorithms, that is, algorithms that work for any group and do not use any special properties of the group at hand. We prove an exponential lower bound on the generic complexity of root extraction, even if the algorithm can choose the "public exponent" itself. In other words, both the standard and the strong RSA assumption are provably true w.r.t. generic algorithms. The results hold for arbitrary groups, so security w.r.t. generic attacks follows for any cryptographic construction based on root extracting. As an example of this, we revisit Cramer-Shoup signature scheme [CS99]. We modify the scheme such that it becomes a generic algorithm. This allows us to implement it in RSA groups without the original restriction that the modulus must be a product of safe primes. It can also be implemented in class groups. In all cases, security follows from a well defined complexity assumption (the strong root assumption), without relying on random oracles, and the assumption is shown to be true w.r.t. generic attacks.

1 Introduction

The well known RSA assumption says, informally speaking, that given a large RSA modulus N, exponent e and $x \in Z_N^*$, it is hard to find y, such that $y^e = x \bmod N$. The strong RSA assumption says that given only N, x it is hard to find any root of x, i.e., to find $y, e > 1$ such that $y^e = x \bmod N$. Clearly the second problem is potentially easier to solve, so the assumption that it is hard is potentially stronger.

Both these assumptions generalize in a natural way to any finite Abelian group: suppose we are given a description of some large finite Abelian group G allowing us to represent elements in G and compute inversion and multiplication in G efficiently. For RSA, the modulus N plays the role of the description. Another example is class groups, where the discriminant Δ serves as the description. Then we can define the *root assumption* which says that given $x \in G$ and number $e > 1$, it is hard to find y such that $y^e = x$. The *strong root assumption* says that given x, it is hard to find $y, e > 1$ such that $y^e = x$.

Clearly, it is essential for these assumptions to hold, that the order of G is hard to compute from the description of G. It must also be difficult to compute discrete logarithms in G. Otherwise, root extraction would be easy: we would be able to find a multiple M of the order of the element x, and then computing $x^{e^{-1} \bmod M}$ would produce the desired root (if $\gcd(e, |G|) = 1$).

[⋆] This is an extended version of what is to appear in Eurocrypt 2002.

[⋆⋆] Part of the work done while visiting IBM Zurich Research Laboratory.

[⋆ ⋆ ⋆] Basic Research in Computer Science,
Centre of the Danish National Research Foundation.

In this paper, we study the complexity of root finding for *generic* algorithms, that is, algorithms that work in any group because they use no special properties of the group G and only use the basic group operations for computing in G. The generic complexity of discrete logarithms was showed to be exponential by Nechaev [Nec94] and Shoup[Sho97] who introduced a broader model for generic algorithms. In both models, however, the algorithm knows the order of the group, which means we cannot use that model for studying the (strong) root assumption as we defined it here [1]. Generic algorithms based on those models were studied and discussed in [MW98a,MW98b,SJ99,Fis00,SJ00,Sch02]. A similar concept of algebraic algorithms was introduced in [BV98].

We propose a new model, in which the group and its order are hidden from the algorithm (as in RSA and class groups setting). More precisely, in our model the group is chosen at random according to a known probability distribution D, and the algorithm only knows that the group order is in a certain interval. For instance, if D is chosen to be the output distribution produced by an RSA key generation algorithm, then this models generic attacks against RSA. But the same model also incorporates generic attacks against schemes based on class groups (we elaborate later on this).

We then show that if D is a so called *hard* distribution, then both the root and strong root assumptions are true for generic algorithms, namely we show exponential lower bounds for both standard and strong root extraction. Roughly speaking, D is hard if, when you choose the group order n according to D, then the uncertainty about the largest prime factor in n is large (later in the paper, we define what this precisely means). For instance, if the distribution is such that the largest prime factor in n is a uniformly chosen $k+1$-bit prime, then, ignoring lower order contributions, extracting roots with non-negligible probability requires time $\Omega(2^{k/3})$ for a generic algorithm. More precise bounds are given later in the paper. Our proof technique resembles that of Shoup [Sho97], however, we need some new ideas in order to take advantage of the uncertainty about group order (which is known and fixed in Shoup's case).

Thus the distribution of the order is the only important factor. Knowledge of the group structure, such as the number of cyclic components, does not help the algorithm. Standard RSA key generation produces distributions that are indeed hard in our sense, so this means that both the standard and the strong RSA assumption are provably true w.r.t. generic algorithms. The results hold for arbitrary groups, so security w.r.t. generic attacks follows for any cryptographic construction based solely on root extracting being hard.

As an example of this, we revisit the Cramer-Shoup signature scheme [CS99]. We modify it such that it becomes a generic algorithm. This allows us to implement it in RSA groups without the original restriction that the modulus must be a product of safe primes. It can also be implemented in class groups. Here, however, we need a "flat tree" signature structure to make the scheme be efficient.

In all cases, security follows from a well defined complexity assumption (the strong root assumption), without relying on random oracles (in contrast to Biehl et al. [BBHM00]).

[1] We note that it *is* possible to study the generic complexity of root finding in Shoups model, but in this setting the problem can only be hard if the "public exponent" is not relatively prime to the order of the group. This was done by Maurer and Wolf [MW98a]. They show that no efficient generic algorithm can compute p'th roots efficiently if p^2 divides the order of the group G and p is a large prime.

We stress that the security proof is general, i.e., if our complexity assumption holds, then the scheme is secure against *all polynomial-time attacks*, not just generic ones. In addition, however, our lower bounds imply that the assumption is true w.r.t. generic attacks.

Before discussing the meaning and significance of our results, we note that one must always be careful in estimating the value of a generic lower bound. This has been demonstrated recently where a server aided RSA protocol proposed in [MKI88] was proved generically secure in [MW98b], but was later broken in [NS01]. This is unrelated to our results because neither the generic proof nor the break applied to root extraction, but to the problem of breaking a particular protocol. We believe this demonstrates that a generic lower bound for a very specific and not so well-studied problem is of highly questionable value: the problem may turn out to be very easy after all, or to be so specialised that there are no variants of it for which generic solutions are the best. From this point of view, it seems reasonable to consider generic lower bounds for root extraction.

Nevertheless, we have of course NOT proved that RSA is hard to break, or that root finding in class groups is hard: for both problems, there are non-generic algorithms known that solve the problem in sub-exponential time. It must also be mentioned that while there are groups known for which only generic algorithms seem to be able to find discrete logs (namely elliptic curve groups), similar examples are not known for root extraction. Despite this, we believe that the results are useful and interesting for other reasons:

- They shed some light on the question whether the strong RSA assumption really is stronger than the standard RSA assumption: since we show exponential generic lower bounds for both problems, it follows that this question must be studied separately for each type of group: only an algorithm directed specifically against the group at hand can separate the two problems.
- They give extra credibility to the general belief that the best RSA key generation is obtained by choosing the prime factors large enough and as randomly as possible, the point being that this implies that $\phi(n)$ contains at least one large and random prime factor, and so the distribution of the group order is hard in our sense. Note that a generic algorithm, namely a straightforward variant of Pollards $p-1$ method, will be able to find roots if all primes factors of $\phi(n)$ are too small. When $\phi(n)$ contains large prime factors this attack is trivially prevented, but our results say that in fact all generic attacks will fail.
- The fact that no examples are known of groups where only generic root finding works, does not mean that such examples do not exist. We hope that our results can motivate research aimed at finding such examples.
- The generic point of view allows us to look at constructions such as the Cramer-Shoup signature scheme in an abstract way, and "move" them to other groups. The generic security of the scheme will be automatically inherited, on the other hand the real, non-generic complexity of root finding may be completely different in different groups. Therefore, having a construction work in arbitrary groups makes it more robust against non-generic attacks.

1.1 Open Problems

It can be argued that the RSA example can be more naturally considered as a ring than as a group. One may ask if root extraction is also hard for generic algorithms allowed to exploit the full ring structure, i.e., it may do additions as well as multiplications. Boneh and Lipton [BL96] have considered this problem for fields, in a model called black-box fields, and have shown that in this model (where the algorithm knows the cardinality of the field) one cannot hide the identity of field elements from the algorithm using random encodings: the algorithm will be able figure out in subexponential time which element is hidden behind an encoding, and so exponential lower bounds in this generic model cannot be shown. The black-box field model can easily be changed to a black-box ring model. But if we do not give the algorithm the order of the multiplicative group of units in the ring, it is unclear whether the results of Boneh and Lipton [BL96] extend to black-box rings, and on the other hand also unclear if our lower bound still holds.

2 Lower Bound on Generic Root Extraction Algorithms

2.1 Model

In our model, we have public parameters B, C and D. Here, B, C are natural numbers and D is a probability distribution on Abelian groups with order in the interval $]B, B+C]$.

Let G be a finite abelian group of order n chosen according to D and let S denote the set of bit strings of cardinality at least $B+C$. We define *encoding function* as an injective map σ from G into S.

A generic algorithm A on S is a probabilistic algorithm which gets as input an *encoding list* $(\sigma(x_1), \dots .\sigma(x_k))$, where $x_i \in G$ and σ is an encoding function of G on S. Note that unlike in Shoups' model [Sho97] this algorithm does not receive the order n of the group as a part of its input.

The algorithm can query a *group oracle* $\mathcal{O}$, specifying two indices i and j from the encoding list, and a sign bit. The oracle returns $\sigma(x_i \pm x_j)$ according to the given sign bit, and this bit string is appended to the encoding list.

The algorithm can also ask the group oracle $\mathcal{O}$ for a random element. Then the oracle chooses a random element $r \in_R G$ and returns $\sigma(r)$, which is appended to the encoding list. After its execution the algorithm returns a bit string denoted by $A(\sigma, x_1, \dots, x_k)$.

Note that this model forces the algorithm to be generic by hiding the group elements behind encodings. In the following we will choose a random encoding function each time the algorithm is executed, and we will show that root extraction is hard for most encoding functions.

2.2 Lower Bound

The distribution D induces in a natural way a distribution D_p over the primes, namely we choose a group of order n according to D and look at the largest prime factor in n. We let $\alpha(D)$ be the maximal probability occurring in D_p. One can think of $\alpha(D)$ as a measure for how hard it is to guess the largest prime factor of n, since clearly the best strategy is to guess at some prime that has a maximal probability of being chosen. We also need to measure how large p can be expected to be. So for an integer M, let $\beta(D, M)$ be the probability that $p \leq M$. As we shall see, a good distribution D (for which root extraction is hard) has small $\alpha(D)$ and small $\beta(D, M)$, even for large values of M [2]. Since, as we shall see, the only important property of D is how the order of the group is distributed, we will sometimes in the following identify D with the distribution of group orders it induces.

First we state a number of observations.

Lemma 1. *Let n be a number chosen randomly from the interval $]B, B+C]$ according to the probability distribution D and let p be its biggest prime divisor. Let a be any fixed integer satisfying $|a| \leq 2^m$. Then the probability that $p|a$ is at most*

$$m\alpha(D).$$

Proof. There are at most m prime numbers dividing a, each of these can be the largest prime factor in n with probability at most $\alpha(D)$.

The following lemma was introduced by Shoup [Sho97]:

Lemma 2. *Let p be prime and let $t \geq 1$. Let $F(X_1, \ldots, X_k) \in \mathbb{Z}/p^t[X_1, \ldots, X_k]$ be a nonzero polynomial of total degree d. Then for random $x_1, \ldots, x_k \in \mathbb{Z}/p^t$, the probability that $F(x_1, \ldots, x_k) = 0$ is at most d/p.*

Lemma 3. *Let $0 < M < B$ and $F(X_1, \ldots, X_k) = a_1X_1 + \cdots + a_kX_k \in \mathbb{Z}[X_1, \ldots, X_k]$ be a nonzero polynomial, whose coefficients satisfy $|a_i| \leq 2^m$. Let the integer n be chosen randomly from the range $]B, B+C]$ according to the probability distribution D. If p is the biggest prime divisor of n and $t \geq 1$, then for random $x_1, \ldots, x_k \in \mathbb{Z}/p^t$, the probability that $F(x_1, \ldots, x_k) = 0$ in Z/p^t is at most*

$$m\alpha(D) + \beta(D, M) + \frac{1}{M},$$

for any M.

Proof. Wlog we can assume that $a_1 \neq 0$. Let E be the event that $F(x_1, \ldots, x_k) = 0$, and A the event that $p > M$ and $p \nmid a_1$. We have $P(E) = P(E, \neg A) + P(E, A) \leq P(\neg A) + P(E|A)$. By lemma 1, and by definition of $\beta(D, M)$ we have that

$$P(\neg A) \leq m\alpha(D) + \beta(D, M).$$

On the other hand, assuming that $p \nmid a_1$ (and $p > M$), we can consider $F(X_1, \ldots, X_k)$ as a nonzero linear polynomial in the ring $\mathbb{Z}/p^t[X_1, \ldots, X_k]$. Let $x_1, \ldots, x_k$ be chosen at random in $\mathbb{Z}/p^t$. Then by lemma 2 we have

$$P(E|A) \leq \frac{1}{p} < \frac{1}{M}.$$

[2] The two measures $\alpha(D), \beta(D, M)$ are actually related, we elaborate below

Now we are able to prove the main theorem.

Theorem 1. *Let a probability distribution D on the range $]B, B+C]$ be given. Let $S \subset \{0,1\}^*$ be a set of cardinality at least $B+C$ and A a generic algorithm on S that makes at most m oracle queries. Let M be any number such that $2m \leq M$. Suppose A is an algorithm for solving the strong root problem, that is, A gets an element from S as input and outputs a number e and an element from S. Assume* $\log e \leq v$[3]*.*

Now choose a group G according to D and let n be its order. Further choose $x \in G$ and an encoding function σ at random. Then the probability that $A(\sigma, x) = e, \sigma(y)$ where $e > 1$ and $ey = x$ is at most

$$(m^3 + m^2 + mv + 2v + m)\alpha(D) + (m^2 + m + 2)\beta(D, M) + (m^2 + m + 3)/M + \frac{m^2}{B}.$$

Proof. First observe that A may output a new encoding from S that it was not given by the oracle, or it may output one of the encodings it was given by the oracle. Without loss of generality, we may assume that A always outputs a new encoding $s \in S$, if we define that A has success if this new encoding represents an e'th root of x, or if *any* of the encodings it was given represents such a root.

We define a new group oracle $\mathcal{O}'$ that works like $\mathcal{O}$, except that when asked for a random element, it chooses at random among elements for which the algorithm does *not* already know encodings. The executions of A that can be produced using $\mathcal{O}'$ are a subset of those one can get using $\mathcal{O}$. For each query, the probability that $\mathcal{O}$ chooses a "known" element is at most (number of encodings known)/(group order). Since at most one new encoding is produced by every oracle quiry, this probability is at most m/n. Hence the overall probability that $\mathcal{O}$ generates something $\mathcal{O}'$ could not is at most $m^2/n \leq m^2/B$. It follows that the success probability of A using $\mathcal{O}$ is at most the sum of the success probability of A using $\mathcal{O}'$ and m^2/B.

To estimate the success probability of A using $\mathcal{O}'$, we will simulate the oracle $\mathcal{O}'$ as it interacts with the algorithm. However, we will not choose the group G or x until after the algorithm halts. While simulating, we keep track of what values the algorithm has computed, as follows: Let $X, Y_1, \ldots, Y_m$ be indeterminants. At any step, we say that the algorithm has computed a list $F_1, \ldots, F_k$ of linear integer polynomials in variables $X, Y_1, \ldots, Y_m$. The algorithm knows also a list $\sigma_1, \ldots, \sigma_k$ of values in S, that the algorithm "thinks" encodes corresponding elements in G. When we start, $k = 1$; $F_1 = X$ and σ_1 is chosen at random and given to A. Whenever the algorithm queries two indices i and j and a bit sign, we compute $F_{k+1} = F_i \pm F_j \in \mathbb{Z}[X, Y_1, \ldots, Y_m]$ according to the bit sign. If $F_{k+1} = F_l$ for some ℓ with $1 \leq l \leq k$, we define $\sigma_{k+1} = \sigma_l$; otherwise we choose σ_{k+1} at random from S distinct from $\sigma_1, \ldots, \sigma_k$.

When the algorithm asks for a random element from the group G, we choose at random σ_{k+1} from $S \setminus \{\sigma_1, \ldots, \sigma_k\}$. We define $F_{k+1} = Y_k$.

When the algorithm terminates, it outputs an exponent e, such that $\log e \leq v$ and $e > 1$ (and also outputs some new element s in S).

[3] The parameter v is at most the running time of A. For example depending on the setting we could define it to be constant or related to m (like $v = O(m)$).

We then choose at random a group G according to the probability distribution D and hence also order n from the interval $]B, B+C]$.

Let p be the biggest prime dividing n. We know that for some integer $r > 0$

$$G \cong \mathbb{Z}_{p^r} \times H.$$

We choose at random $x, y_1, \ldots, y_m \in_R G$, which is equivalent to random independent choices of $x', y'_1, \ldots, y'_m \in_R \mathbb{Z}_{p^r}$ and $x'', y''_1, \ldots, y''_m \in_R H$.

We say that *the algorithm wins* if either

- $F_i(x', y'_1, \ldots, y'_m) \equiv F_j(x', y'_1, \ldots, y'_m) \bmod p^r$ and $F_i \neq F_j$ for some $1 \leq i \neq j \leq m+1$, or:
- The new encoding s output by A is an e'th root of x, or:
- $eF_i(x', y'_1, \ldots, y'_m) \equiv x' \bmod p^r$ for some $1 \leq i \leq m+1$.

To estimate the probability that the algorithm wins, we simply estimate the probability that each of the above three cases occur.

For this first case, recall that $F_i(X, Y_1, \ldots, Y_m) = a_{i,0}X + a_{i,1}Y_1 + \cdots + a_{i,m}Y_m$, where $F_1(X, Y_1, \ldots, Y_m) = X$. Since at each oracle query we could only either introduce a new polynomial Y_k, add or subtract polynomials, then $|a_{i,j}| \leq 2^m$ for $i = 1, \ldots, m+1, j = 0, \ldots, m$. Lemma 3 implies that $F_i(x', y'_1, \ldots, y'_m) \equiv F_j(x', y'_1, \ldots, y'_m) \bmod p^r$ for given $i \neq j$, $F_i \neq F_j$ with probability at most $m\alpha(D) + \beta(D, M) + \frac{1}{M}$. Since we have at most m^2 pairs $i \neq j$, the probability that $F_i(x', y'_1, \ldots, y'_m) \equiv F_j(x', y'_1, \ldots, y'_m) \bmod p^r$ for some $i \neq j$, $F_i \neq F_j$ is at most

$$m^2 \left(m\alpha(D) + \beta(D, M) + \frac{1}{M} \right).$$

For the second case, first observe that the probability that $p|e$ or that $p < M$ is at most $v\alpha(D) + \beta(D, M)$, by lemma 1. On the other hand, assuming that $p \nmid e$ and $p \geq M$, a random group element different from the at most m elements the algorithm has been given encodings of is an e'th root of x with probability at most $\frac{1}{p-m} \leq \frac{1}{M-m} \leq \frac{2}{M}$. We conclude: the new encoding output by A represents an e'th root of x with probability at most

$$v\alpha(D) + \beta(D, M) + 2/M.$$

Finally let us consider the event $eF_i(x', y'_1, \ldots, y'_m) \equiv x' \bmod p^r$ for given $1 \leq i \leq m+1$. Since $F_i(X, Y_1, \ldots, Y_m)$ is an integer polynomial and $e > 1$, then $eF_i(X, Y_1, \ldots, Y_m) \neq X$ in $\mathbb{Z}[X, Y_1, \ldots, Y_m]$. We know that $\log e \leq v$ and coefficients $a_{i,j}$ of the polynomial F_i satisfy the condition $|a_{i,j}| \leq 2^m$. Therefore by lemma 3 the probability that $eF_i(x', y'_1, \ldots, y'_m) \equiv x' \bmod p^r$ is at most $(v+m)\,\alpha(D) + \beta(D, M) + \frac{1}{M}$. Hence the event, that for some $1 \leq i \leq m+1$ the equivalence $eF_i(x', y'_1, \ldots, y'_m) \equiv x' \bmod p^r$ holds, happens with the probability

$$(m+1)\left((v+m)\,\alpha(D) + \beta(D, M) + \frac{1}{M} \right).$$

Summarizing, the probability that A wins when we simulate $\mathcal{O}'$ is at most

$$(m^3 + m^2 + mv + 2v + m)\alpha(D) + (m^2 + m + 2)\beta(D, M) + (m^2 + m + 3)/M.$$

Given this bound on the probability that A wins the simulation game, we need to argue the connection to actual executions of A (using $\mathcal{O}'$). To this end, note that since the only difference between simulation and execution is the time at which the group, group elements and encodings are chosen, the event that

$$F_i(x', y'_1, \ldots, y'_m) \equiv F_j(x', y'_1, \ldots, y'_m) \bmod p^r \text{ and } F_i \neq F_j \text{ for some } 1 \leq i \neq j \leq m+1$$

is well defined in both scenarios. Let BAD_{sim}, respectively BAD_{exec} be the events that this happens in simulation, respectively in an execution. We let a *history* of execution or simulation be the choice of group and group elements followed by the sequence of oracle calls and responses, followed by A's final output. We then make the following

Claim: Every history in which BAD_{sim} does not happen in simulation of $\mathcal{O}'$ occur with the same probability as history in which BAD_{exec} does not happen in execution of $\mathcal{O}'$. In particular, $P(BAD_{sim}) = P(BAD_{exec})$.

Recall that we defined that A wins the simulation game if BAD_{sim} occurs, or A finds a root. Therefore the claim clearly implies that our bound on the probability that A wins the simulation game is also a bound on A's success probability in a real execution.

To show the claim, it is sufficient to prove it for every fixed value of A's random choices and every fixed choice of group and group elements. This means that the choice of encodings is the only source of randomness. Consider some partial execution history in which a bad event has not yet happened, and A has just made oracle call number i. Since no bad event has happened yet, this means that the oracle call just made either defines a polynomial that was already defined earlier, or it defines a new polynomial, but the corresponding group element is different from all those defined by earlier polynomials. This means that both in simulation and in execution, the response will either be an encoding given before, or a new randomly chosen value different from all earlier encodings. Therefore the probability that seeing this encoding will make A cause a bad event with the next oracle call is exactly the same in execution as in simulation. This implies the first part of the claim, the second part follows easily from the first.

We now consider the case, where we have a family of values of the parameters B, C, D, i.e., they are functions of a security parameter k. As usual we say that a function of k is negligible if it is at most $1/p(k)$ for any polynomial $p()$ and all large enough k.

Definition 1. *Let $\{D(k) |\ k = 1, 2, ...\}$ be a family of probability distributions, where $D(k)$ ranges over Abelian groups of order in the interval $]B(k), B(k) + C(k)]$. The family is said to be* hard *if $\alpha(D(k))$ and $\frac{1}{B(k)}$ are negligible in k.*

In the following, we will sometimes write just D in place of $D(k)$ when the dependency on k is clear. We can observe:

Fact 1 *If $\{D(k) |\ k = 1, 2, ...\}$ is a hard family, then there exists $M(k)$, such that $\beta(D(k), M(k))$ and $\frac{1}{M(k)}$ are negligible in k.*

Proof. Let $M(k) = \frac{1}{\sqrt{\alpha(D(k))}}$. Clearly $\frac{1}{M(k)} = \sqrt{\alpha(D(k))}$ is negligible.

Now let n be the order of a group chosen according to $D(k)$. What is the probability that the biggest prime factor of n is smaller than $M(k)$? We have at most $M(k)$ prime factors smaller than $M(k)$. Each can be chosen with probability at most $\alpha(D(k))$. Therefore

$$\beta(D(k), M(k)) \leq M(k)\alpha(D(k)) = \frac{1}{\sqrt{\alpha(D(k))}}\alpha(D(k)) = \sqrt{\alpha(D(k))},$$

which is negligible.

Corollary 1. *Let the family $\{D(k)|\ k = 1, 2, ...\}$ be hard. We choose randomly an abelian group G of order n according to the distribution $D(k)$. Let $x \in G$ be chosen at random. Consider any probabilistic polynomial time generic algorithm A. The probability that A given x outputs a natural number $e > 1$ and an encoding of an element $y \in G$ such that $y^e = x$ is negligible.*

Proof. Let M be such that $\alpha(D)$, $\beta(D, M)$ and $\frac{1}{M}$ are negligible in k. Let m be the number of group queries made by the algorithm A. Then m and $\log e$ are polynomial in k, so clearly $2m < M$ for all large enough values of the security parameter. It follows from Theorem 1 that the probability that A succeeds is negligible.

One may note that if $\alpha(D(k))$ and $1/B(k)$ are exponentially small in k (as it is in the concrete examples we know), say $\alpha(D(k)) \leq 2^{-ck}$ for a constant $c > 0$ and $B(k) \geq 2^{dk}$ for a constant $d > 0$. Suppose the running time of a generic algorithm A is $O\left(2^{c'k}\right)$ where $c' < \min(c/4, d/2)$. Then it follows from the concrete expression in Theorem 1 and the proof of Fact 1 that A has negligible success probability. Thus for such examples, we get an exponential generic lower bound.

Assume that $D(k)$ is such that the largest prime factor in the order is a uniformly chosen $k+1$-bit prime, then certainly $B(k) \geq 2^k$, and also by the prime number theorem $\alpha(k) \approx 2^{-k+\log k} \ln 2$. If we set $M = 2^k$ then clearly $\beta(D, M) = 0$. Therefore by Theorem 1 if we ignore lower order contributions we get that root extraction requires time $\Omega(2^{k/3})$.

We present some proven or conjectured examples of hard families of distributions below in next subsections.

Applying the results to the standard root problem. What we have said so far concerns only the strong root assumption. For the standard root assumption, the scenario is slightly different: A group G and a public exponent $e > 1$ are chosen according to some joint distribution. Then e and an encoding of a random element x is given as input to the (generic) algorithm, which tries to find an e'th root of x. Clearly, if we let D be the distribution of G given e, Theorem 1 can be applied to this situation as well, as it lower bounds the complexity of finding *any* root of x.

Now consider a family of distributions as above $\{D(k)|\ k = 1, 2, ...\}$, where each $D(k)$ now outputs a group G and an exponent e, let $D(k, e)$ be the distribution of G given e, and $]B(k, e), B(k, e) + C(k, e)]$ be the interval in which $|G|$ lies, given e. The family is said to be *hard* if $Max_e(\alpha(D(k,e)))$ and $Max_e(\frac{1}{B(k,e)})$ are negligible in k, where the maximum is over all e that can be output by $D(k)$. One can then show in a straightforward way a result corresponding to Corollary 1 for the standard root problem.

2.3 RSA

For the RSA case, it is clear that any reasonable key generation algorithm would produce hard distributions because it is already standard to demand from good RSA key generation that the order of the group produced contains at least one large and random prime factor.

Furthermore, the fact that a public exponent e with $(e,|G|)=1$ is given does not constrain seriously the distribution of the group, i.e., it remains hard. Consider, for instance, the distribution of random RSA moduli versus random RSA moduli where 3 is a valid public exponent. Since essentially half of all primes p satisfy that 3 does not divide $p-1$, the uncertainty about the largest prime factor in the group order remains large.

2.4 The Uniform Distribution

Assume our distribution D is such that the group order n is chosen from the interval $]B, B+C]$ according to the uniform distribution, and such that both B and C are at least 2^k, where k is the security parameter. We prove that this uniform distribution is *hard.*

Let $u=\sqrt{k}$. Wlog we can assume that u is a natural number. Let $M=2^u$. Clearly $\frac{1}{M}$ is negligible in k.

We prove that in this case $\alpha(D)$ and $\beta(D,M)$ are also negligible.

Fact 2 *$\beta(D,M)$ is negligible in k.*

Proof. Let $\epsilon>0$ be fixed. It is easy to see that

$$1 \le u \le \exp\{u^{\frac{3}{5}-\epsilon}\}$$

for almost all k. Since $u^2 \le \log(B+C)$, we know that

$$u \le \exp\left\{\left(\log\left((B+C)^{\frac{1}{u}}\right)\right)^{\frac{3}{5}-\epsilon}\right\}$$

for almost all k. It follows from [Mor93, Chapter 1] that the probability that the biggest prime divisor p of n is at most $M=2^u$ (recall that $M=2^u \le (B+C)^{\frac{1}{u}}$) is at most

$$\rho(u)\left(1+O_\epsilon\left(\frac{\log(u+1)}{\log\left((B+C)^{\frac{1}{u}}\right)}\right)\right) \le \rho(u)\left(1+O_\epsilon\left(\frac{\log(u+1)}{u}\right)\right).$$

Moreover by [Mor93, Chapter 2,Lemma 3]

$$\rho(u) \le \frac{1}{\Gamma(u+1)} = \frac{1}{u!}.$$

Hence the probability that the biggest prime divisor p of n is at most $M=2^u$ is negligible.

Fact 3 *With the above definition of D, $\alpha(D)$ is negligible in k.*

Proof. Let p be the biggest prime divisor of n. By the Fact 2 the probability that $p \leq M$ is negligible. Therefore we can assume that $p > M$.

We have C numbers in the interval $]B, B+C]$. The biggest prime divisor p of n can divide at most $1 + C/p$ numbers from the interval $]B, B+C]$, hence it can be chosen with the probability at most

$$\frac{1 + C/p}{C} = \frac{1}{C} + \frac{1}{p} < \frac{1}{2^k} + \frac{1}{M} \ .$$

2.5 Class Groups

Buchmann and Williams [BW88] proposed a first crypto system based on class groups of imaginary quadratic orders (IQC). We describe briefly class groups and their application in cryptography.

Let Δ be a negative integer such that $\Delta \equiv 0, 1 \bmod 4$. The *discriminant* Δ is called *fundamental* if $\frac{\Delta}{4}$ or Δ is square free for $\Delta \equiv 0 \bmod 4$ or $\Delta \equiv 1 \bmod 4$, respectively. Given Δ it is possible to construct an abelian finite group called *class group* and denoted by $Cl(\Delta)$. The order of the class group is denoted by $h(\Delta)$ and called *class number.* We denote the cardinality of the odd part of a class group $Cl(\Delta)$ by $h_{\text{odd}}(\Delta)$.

There is no known efficient algorithm to compute class number $h(\Delta)$ if the discriminant Δ is fundamental. Moreover the fastest known algorithm (Jacobson [Jac99]) to compute class number is a variant of the MPQS (Multiple Polynomial Quadratic Sieve) factoring algorithm. It is asymptotically much slower (empirical data suggests $L_{|\Delta|}[\frac{1}{2}, 1]$) than the GNFS (General Number Field Sieve) algorithm for factoring integers (expected running time proportional to $L_n[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}]$).

On the one hand we can reason that smaller numbers can be used in class group arithmetic in comparison with RSA. We can potentially have faster crypto systems based on class groups of imaginary quadratic orders.

On the other hand it can be argued that we don't know $L_{|\Delta|}[\frac{1}{3}, c]$ algorithm for the computation of discrete logarithms in class groups, because it has not been so extensively researched as the factorization problem. Note also that Hühnlein and Takagi [HT99] proposed $L_{|\Delta|}[\frac{1}{3}, c]$ algorithm for the special case of class groups of totally non-maximal imaginary quadratic orders.

Note that due to Gauss, Shanks and Lagarias [Lag80,HMT99] there is an efficient algorithm which given the factorization of Δ computes square roots in $Cl(\Delta)$. Therefore only odd exponents should be considered hard for root extraction in $Cl(\Delta)$ if the prime factors of Δ are known.

We know that for any fixed $\epsilon > 0$

$$\sqrt{|\Delta|}^{1-\epsilon} \leq h(\Delta)$$

for sufficiently large Δ. Also

$$h(\Delta) \leq \frac{2}{\pi}\sqrt{|\Delta|}\left(1 + \log\left(\frac{2\sqrt{\Delta}}{\pi}\right)\right)$$

for sufficiently large Δ.

Moreover under the Extended Riemann Hypothesis

$$\frac{1+o(1)}{c_2 \log\log|\Delta|}\sqrt{|\Delta|} \leq h(\Delta) < (1+o(1))\, c_3\sqrt{|\Delta|}\log\log|\Delta|$$

where $c_2 = \frac{12e^\gamma}{\pi} \approx 6.8$ and $c_3 = \frac{2e^\gamma}{\pi} \approx 1.134$.

Buell [Bue84] has analyzed class groups with discriminants $\Delta = -pq$ and $\Delta = -4pq$, where p and q are primes. Based on statistical evidence Buell offers the conjecture that for a small odd prime p, p^n divides class numbers with probability $\frac{1+\frac{1}{p}}{p^n}$. This can indicate that class numbers are smoother than random numbers.

Hamdy and Möller [HM00] analyze the smoothness probabilities of class numbers based on heuristics of Cohen and Lenstra [CHL84] and Buell [Bue84]. They estimate that for $M \approx \frac{h_{\text{odd}}(\Delta)}{h(\Delta)}c\sqrt{|\Delta|}$ where $c = 0.461559\ldots$, the probability that the biggest prime divisor of $h(\Delta)$ is smaller than $M^{\frac{1}{u}}$ is at most $\rho(u)\ln\ln M$. Hence to ensure that $h(\Delta)$ has a big prime factor we need $h_{\text{odd}}(\Delta)$ to be big.

Hamdy and Möller provide two examples of discriminants with big $h_{\text{odd}}(\Delta)$. The first example is $\Delta = -p$ where $p \equiv 3 \bmod 4$ is prime (in this case $h_{\text{odd}}(\Delta) = h(\Delta)$). The second example is $\Delta = -8pq$ where p and q are primes such that $p \equiv 1 \bmod 8$, $p + q \equiv 8 \bmod 16$ and the Legendre symbol $\left(\frac{p}{q}\right)$ is equal -1 (then $h_{\text{odd}}(\Delta) = \frac{h(\Delta)}{8}$).

We assume that a discriminant Δ is chosen randomly from some interval (under the restriction in above examples) and that results in a distribution D of class numbers. In provided examples we have that $\beta(D, M)$ and $\frac{1}{M}$ are negligible ($\frac{1}{B}$ is trivially negligible).

Buell [Bue84] observes that class numbers are not evenly distributed, but obey some sort of gamma-distribution. Recall that the density of gamma random variable with parameters $\lambda, t > 0$ is $f(x) = \frac{\lambda e^{-\lambda x}(\lambda x)^{t-1}}{\Gamma(t)}$ for $x \geq 0$. The expectation of gamma random variable is $E(X) = \frac{t}{\lambda}$ and the variance $var(X) = \frac{t}{\lambda^2}$.

The class number is of size about the square root of the discriminant, so even if we restrict the discriminant to be $-p$ with p prime (or $-8pq$), it seems that number of possible discriminants is much bigger than the number of possible class numbers. Perhaps this, together with Buell's observation about the gamma distribution can be used to argue that the class numbers produced are not too far from uniform in some interval, and hence the distribution D is hard. This requires further investigation. In any case, we believe it is reasonable to conjecture that the distributions induced by choosing discriminant as described above are indeed hard.

3 Generic Cryptographic Protocols

Based on our model of generic adversary, we look at the notion of generic cryptographic protocols. Such protocols do not exploit any specific properties of group element representation. They are based on a simple model of finite abelian group with unknown order.

One advantage of this approach is an easy application of generic protocols in any family of groups of difficult to compute order. As a first example we note the family of RSA groups (without necessarily restricting the RSA modulus to a product of safe primes). Another example is class groups.

This generic approach has been used earlier: Shoup [Sho97] proved the security of Schnorr's identification scheme [Sch91] in the generic model with public order of the group. Biehl et al. [BBHM00] introduced some generic cryptographic protocols and applied them in class groups.

In this section we propose a generic signature scheme which can be seen as an abstraction of Cramer-Shoup signature scheme [CS99]. We will show security of this protocol against arbitrary attacks, given certain intractability assumption (strong root assumption), which we have proven in our generic model. This immediately implies provable security with respect to generic attacks.

3.1 Protocol

For this signature scheme, we assume that we are given D, a probability distribution over Abelian groups with order in the interval $]B, B+C]$. We also assume for this version of the scheme that one can efficiently choose a group according to D, such that the order of the group is known (the assumption about known order is removed later). Moreover, we assume a collision-resistant hash function H whose output is a natural number smaller than $\frac{B}{2}$.

Key Generation. We choose G of order n according to the probability distribution D. A description of the group G is announced, but its order remains hidden (for RSA, for instance, this amounts to publishing the modulus).

We choose randomly elements $h \in_R G$ and $x \in_R < h >$(let $< h >$ denote the cyclic subgroup generated by h). Let e' be a prime satisfying $B + C < e' < 2\,(B+C)$.

The public key is (h, x, e').

The private key is n.

Signing Protocol

1. When a message M (it can be a bit string of arbitrary size) is requested to be signed, a random prime $e \neq e'$ is chosen such that $B + C < e < 2\,(B+C)$. A random element $y' \in_R G$ is chosen.
2. We compute
$$x' = \frac{(y')^{e'}}{h^{H(M)}}$$
and
$$y = \left(xh^{H(x')}\right)^{\frac{1}{e}}.$$
Note that the signer can invert e, since he knows the order n of the group G.
3. The signature is defined to be (e, y, y').

Signature Verification. To verify that (e, y, y') is a signature on a message M, we perform the following steps.

1. We check that e is a prime integer from the interval $]B + C, 2(B + C)[$ and different from e'.
2. We compute $x' = (y')^{e'} h^{-H(M)}$.
3. We verify that $x = y^e h^{-H(x')}$.

3.2 Preliminary Observations

In order to prove security of our protocol first we prove some preliminary observations.

Fact 4 *For any $E > 0$, given natural numbers $e_1, \ldots, e_t$ smaller than E and an element g in a finite abelian multiplicative group G, we are able to compute $g^{\prod_{i \neq j} e_i}$ for all $j = 1, \ldots, t$ using $O(t \log t \log E)$ multiplications in G.*

Proof. Wlog we can assume that $t = 2^r$. We prove by induction on r that we can compute $g^{\prod_{i \neq j} e_i}$ for all $j = 1, \ldots, t$ using at most $C_0 t \log t \log E$ multiplications in G for some constant $C_0 > 0$.

The base case $r = 1$ is obvious.

Suppose that for $t = 2^r$ we can compute $g^{\prod_{i \neq j} e_i}$ for all $j = 1, \ldots, t$ using at most $C_0 t \log t \log E$ multiplications in G.

Assume that we have $2t$ random numbers $e_1, \ldots, e_{2t}$ smaller than $E > 0$. We need to show that we can compute $g^{\prod_{i \neq j} e_i}$ for all $j = 1, \ldots, 2t$ using at most $C_0 2t \log(2t) \log E$ multiplications in G.

Let $f_i = e_{2i-1} e_{2i}$ for all $1 \leq i \leq t$. Observe that $0 < f_i < E^2$. By induction hypothesis we can compute $h_i = g^{\prod_{i \neq j} f_i}$ for all $j = 1, \ldots, t$ using at most

$$C_0 t \log t \log\left(E^2\right) = 2C_0 t \log t \log E$$

multiplications in G.

Note that

$$g^{\prod_{i \neq 2j, 1 \leq i \leq 2t} e_i} = \left(g^{\prod_{i \neq j, 1 \leq i \leq t} f_i}\right)^{e_{2j-1}} = h_j^{e_{2j-1}}$$

and analogously

$$g^{\prod_{i \neq 2j-1, 1 \leq i \leq 2t} e_i} = \left(g^{\prod_{i \neq j, 1 \leq i \leq t} f_i}\right)^{e_{2j}} = h_j^{e_{2j}}.$$

Hence given the numbers $h_1, \ldots, h_t$ computing $g^{\prod_{i \neq j} e_i}$ for all $j = 1, \ldots, 2t$ requires at most $C_1 t \log E$ group multiplications in G for some constant $C_1 > 0$.

Summarizing all our computations required at most

$$2C_0 t \log t \log E + C_1 t \log E = (2C_0 \log t + C_1) t \log E$$

group operations in G. We can assume that $C_1 < 2C_0 \log 2$. Then

$$(2C_0 \log t + C_1) t \log E < (2C_0 \log t + 2C_0 \log 2) t \log E = 2C_0 (\log t + \log 2) t \log E = 2C_0 \log(2t) t \log E.$$

That finishes the induction step.

Fact 5 *Let $g, h \in G$ be such that*

$$g^a = h^b.$$

If $\gcd(a, b) = 1$*, then it is easy to find a b'th root of g.*

Proof. We can execute extended Euclidian algorithm to obtain c and d such that $ac+bd = 1$. We can observe that

$$g = g^{ac+bd} = h^{bc}g^{bd} = \left(h^c g^d\right)^b.$$

Fact 6 *Let $g, h \in G$ be such that*

$$g^a = h^b.$$

If $b \nmid a$ then it is easy to find a nontrivial root of either g or 1*.*

Proof. Let $r := \gcd(a, b)$. If $r = 1$ then it is a straightforward application of fact 5. Assume that $1 < r$. Obviously $\gcd\left(\frac{a}{r}, \frac{b}{r}\right) = 1$. If

$$g^{\frac{a}{r}} = h^{\frac{b}{r}}$$

then by fact 5 we can easily compute $\left(\frac{b}{r}\right)$'th root of g. Otherwise $g^{\frac{a}{r}}h^{\frac{-b}{r}}$ is an r'th root of 1.

3.3 Security Proof

First we state our intractability assumption:

Conjecture 1. We choose randomly an abelian group G of order n, where $n \in]B, B + C]$, according to the probability distribution D. Let $x \in G$ be chosen at random. Consider any probabilistic polynomial time algorithm A. The probability that A, given the description of G and x, outputs $e > 1$ and $y \in G$ such that $y^e = x$ is negligible.

In our generic model, where A knows only D and encodings of group elements, and assuming that D is from a *hard* family, it follows from the results we proved above that this assumption is true.

We now prove that our signature scheme is secure against chosen message attacks. We stress that this security proof holds for any type of polynomial-time attack (not just generic ones), as long as the conjecture holds.

Theorem 2. *The above signature scheme is secure assuming the conjecture 1 and that H is collision-resistant.*

Proof. Suppose that the adversary A is able to forge a new signature (e, y, y') after t signing queries.

We build a simulator of the signing oracle for the adversary. We show that if the adversary is successful in forging, then the simulator can break the strong root assumption (in some cases just the root assumption).

Let group G of order $n \in]B, B+C]$ be chosen randomly according to the distribution D. A description of group G is published, so that both the simulator and the adversary can perform group operations in G. However the order n of the group G is not directly revealed to them (and it should be difficult to compute).

Let z be a random element of group G. The simulator receives the element z. Next a random element $a \in [0, (B+C)^2]$ is chosen. The simulator computes $\omega = z^a$.

Let the i'th signature on message m_i be (e_i, y_i, y'_i) for $1 \leq i \leq t$.

We can observe three types of forgeries:

Type 1 There is $1 \leq j \leq t$ such that $e = e_j$ and $x' = x'_j$.

Type 2 There is $1 \leq j \leq t$ such that $e = e_j$ and $x' \neq x'_j$.

Type 3 There is no $1 \leq j \leq t$ such that $e = e_j$.

Any forged signature is of one of the above types.

The simulation is similar in all the above cases. However some details and analysis vary. We are going to give a detailed simulation for type 1 forgery and sketch the other cases.

Type 1 forgery. The simulator receives also a prime e' such that $B+C < e' < 2(B+C)$.

It generates random distinct primes $e_1, \ldots, e_t$ with $B+C < e_i < 2(B+C)$ and $e_i \neq e'$ for $i = 1, \ldots, t$. The simulator computes

$$h = z^{\prod_i e_i} \text{ and } x = \omega^{\prod_i e_i}.$$

Clearly $x = h^a$.

Since $gcd(n, \prod_i e_i) = 1$ the element h is uniformly distributed in G. Moreover our distribution of x is statistically indistinguishable from the uniform distribution in the subgroup $< h >$.

The public key is

$$(h, x, e').$$

When a message m_i is requested to be signed, the simulator chooses e to be equal e_i and follows the signing protocol. The signature is (e_i, y_i, y'_i). The above procedure requires computing $\left(xh^{H(x'_j)}\right)^{\frac{1}{e_j}}$ for all $1 \leq j \leq t$. We can observe that

$$\left(xh^{H(x'_j)}\right)^{\frac{1}{e_j}} = \omega^{\prod_{i \neq j} e_i} \left(z^{H(x'_j) \prod_{i \neq j} e_i}\right).$$

Hence the fact 4 implies that the whole simulation is using $O(t \log t \log(B+C))$ group operations.

Since the simulated key generation was statistically indistinguishable from the real key generation, the signing was simulated almost perfectly.

Assume that the adversary creates a forgery (e, y, y') of Type 1 on a message m. Hence $e = e_j$ and $x' = x'_j$. It follows that

$$h^{H(m)} x' = \left(y'\right)^{e'},$$
$$h^{H(m_j)} x' = \left(y'_j\right)^{e'}.$$

So

$$h^{H(m_j)-H(m)} = \left(\frac{y'_j}{y'}\right)^{e'}.$$

Recall that H is collision-resistant. It follows that

$$0 \leq H(m) \neq H(m_j) < \frac{B}{2} < e'.$$

Since e' is prime, it follows from fact 5 that we can compute such $v \in G$ that

$$v^{e'} = h = z^{\prod_i e_i}.$$

We know that $\gcd(e', \prod_i e_i) = 1$. Hence the fact 5 implies that we can easily compute $e'th$ root of z. That contradicts the Conjecture 1.

Note that the simulator used $O\left(t \log t \log(B+C)\right)$ group operations and that the simulator is a generic algorithm.

Type 2 forgery. Here (apart from z) the simulator receives a prime r such that $B+C < r < 2\,(B+C)$.

Let e' be random prime number with $B+C < e' < 2\,(B+C)$.

Let $e_j = r$. The simulator generates random distinct primes e_i with $B+C < e_i < 2\,(B+C)$, $e_i \neq e'$ and $e_i \neq r$ for $i = 1, \ldots, t, i \neq j$. We define

$$h = z^{e' \prod_{i \neq j} e_i}.$$

The simulator computes $y_j = \omega^{\prod_{i \neq j} e_i}$.

We choose a random $u \in_R G$ and define

$$x'_j = u^{e'}.$$

Finally we set

$$x = y_j^{e_j} h^{-H(x'_j)} = \omega^{\prod_i e_i} h^{-H(x'_j)} = h^{\frac{e_j a}{e'} - H(x'_j)}.$$

When a message m_i $(i \neq j)$ is requested to be signed, the simulator chooses e to be equal e_i and follows the signing protocol.

Now suppose that a message m_j is requested to be signed. We compute

$$y'_j = \left(x'_j h^{H(m_j)}\right)^{\frac{1}{e'}} = u z^{H(m_j) \prod_{i \neq j} e_i}.$$

It is easy to check that y_j was defined correctly. Indeed

$$\left(x h^{H(x'_j)}\right)^{\frac{1}{e_j}} = \left(y_j^{e_j} h^{-H(x'_j)} h^{H(x'_j)}\right)^{\frac{1}{e_j}} = y_j.$$

Assume that the forger creates a forgery (e, y, y') of Type 2 on a message m. Hence $e = e_j$ and $x' \neq x'_j$. It follows that

$$x = y^{e_j} h^{-H(x')},$$
$$x = y_j^{e_j} h^{-H(x'_j)}.$$

So

$$h^{H(x'_j)-H(x')} = \left(\frac{y_j}{y}\right)^{e_j} .$$

This allows us to compute the e_j'th root of z.

Type 3 forgery. Let $e', e_1, \ldots, e_t$ be distinct random prime numbers chosen from the interval $]B+C, 2(B+C)[$.

The simulator computes

$$h = z^{\prod_i e_i} \text{ and } x = \omega^{\prod_i e_i}.$$

Suppose that the forger creates a valid signature (e, y, y') of Type 3 on a message m. Then

$$y^e = xh^{H(x')} = z^r,$$

where

$$r = \left(\prod_{i=1}^{t} e_i\right) \cdot (a + H(x'))$$

and $\gcd\left(e, \prod_{i=1}^{t} e_i\right) = 1$. Therefore by fact 5 we can compute $\left(\prod_{i=1}^{t} e_i\right)$'th root v of y and since $\gcd\left(n, \prod_{i=1}^{t} e_i\right) = 1$ we have

$$v^e = z^{a+H(x')} \ .$$

Observe that e is a prime greater then the group order n . Let $a = a_1 n + a_2$ where $0 \le a_1 \le \frac{(B+C)^2}{n}$ and $0 \le a_2 < n$. Clearly the distributions of a_2 and a_1 are almost uniform and independent. The only information related with a the adversary could learn could be derived from $\omega = z^a = z^{a_2}$, hence the adversary has almost no information about a_1. Therefore $a_1 n \bmod e$ from the adversary's point of view is almost uniformly distributed among the values $0, 1, 2, \ldots, e-1$ and he has almost no information about it. Thus the probability that $e|(a_1 n + a_2 + H(x'))$ is approximately $\frac{1}{e}$. Hence with overwhelming probability by Fact 5 it is easy to compute the e'th root of z.

Corollary 2. *The above signature scheme is secure against chosen message attack performed by any generic algorithm if we assume that D is a hard probability distribution.*

Proof. We can observe that the simulation in the above proof is clearly a generic algorithm. Moreover we know that the conjecture 1 is provable in generic framework. Hence the thesis of the corollary follows from the theorem 2.

3.4 Efficiency Improvements

First of all we can improve the efficiency of verification process at the expense of additional assumption.

Conjecture 2. We choose randomly an abelian group G of order n, where $n \in]B, B+C]$, according to the probability distribution D. Consider any probabilistic polynomial time algorithm A. The probability that A, given the description of G, outputs $e > 1$ and $y \in G$ such that $y^e = 1$ is negligible.

It is not difficult to see that the above conjecture can be proven with respect to generic algorithms if the probability distribution D is hard, analogously to our lower bound proof in the previous chapter.

Moreover the conjecture is true for some RSA cases. Let the RSA moduli $N = pq$ be such that $\gcd(p-1, q-1) = 2$ (similar discussion appeared in [DF02]). If we assume that e is odd and can be efficiently factorized then given an e'th root of 1 modulo N we can find such $y \bmod N$ and an odd prime p_0 that $y^{p_0} = 1 \bmod N$. So we can deduce that either $y \equiv 1 \bmod p$ or $y \equiv 1 \bmod q$ but not both, which allows us to factor N.

With this additional assumption, we can prove our signature scheme secure without requiring from the verifier checking the primality of exponent e. Let's limit the point 1) in the verification procedure to testing only that e is an odd integer from the interval $]B+C, 2(B+C)[$ and different from e'.

Proof. It is similar to the proof of Theorem 2. Types 1 and 2 forgeries are dealt with in the same way.

Concerning Type 3, we end up with element v and equation

$$v^e = z^{a+H(x')} \quad ,$$

as in the previous proof.

Now since $e > n$ we have that

$$e \nmid n \ . \tag{1}$$

So there are such prime p_0 and integer $r \geq 0$ that $p_0^{r+1} \mid e$ and $p_0^r \mid n$, but $p_0^{r+1} \nmid n$. Therefore $a_1 n \bmod p^{r+1}$ from the adversary's point of view can be any of $0, p^r, 2p^r, \ldots, (p-1)p^r$ and he has almost no information about it. Thus the probability that $p^{r+1} | (a_1 n + a_2 + H(x'))$ is approximately $\frac{1}{p}$. Hence with non-negligible probability $e \nmid (a + H(x'))$. Therefore by fact 6 we can compute a nontrivial root of either z or 1 which contradicts respectevily Conjectures 1 and 2.

That gives us much faster verification process.

Known order. If we are able to choose a random group of order known for us and hidden for others, then we have a reasonably efficient signature scheme, secure assuming our conjectures.

RSA Let us assume that the output length l of hash function $H()$ is much smaller. Simply l must be only big enough to ensure that $H()$ is collision-resistant. For example for current security requirements we can assume that $l = 160$. Let the exponents in our signature scheme be $l+1$-bit exponents as in Cramer-Shoup original scheme.

Note that 161-bit exponents can be easily factorized. In asymptotic terms we cannot say rigorously if $l+1$-bit number can be efficiently factored. But if we choose l so that

$H()$ withstands the birthday paradox as it is done nowadays, then $l+1$-bit numbers can be factored efficiently since we have at disposal subexponential factorization algorithms.

As long as we believe that we can factor efficiently $l+1$-bit numbers if we choose RSA moduli $N = pq$ such that $\gcd(p-1, q-1) = 2$ then Conjecture 2 is true w.r.t. odd exponents e.

Note also that (1) remains true. Therefore our scheme with 161-bit exponents can be proven secure under the strong RSA assumption. The original Cramer-Shoup signature scheme [CS99] can be seen as a special case of this, where the RSA keys are restricted to safe prime products.

Unknown order. If we do not know how to generate a group of known for us order, we cannot use the signing algorithm we specified above. Instead, we can use the method given in the simulation from the proof in order to sign messages. This requires $O(kt \log t)$ group operations, where t is the number of the messages to be signed and k the security parameter. Moreover we need to keep t exponents e in our memory.

Our signature scheme in such form can be applied, for instance, in class groups where we do not know how to generate efficiently a discriminant together with the corresponding class number. However, as t becomes large, this scheme becomes rather inefficient. In the following section we describe a solution to this.

4 Tree Structured Signature Scheme in groups with unknown order

Here, we describe a generic signature scheme, which can be applied, e.g, in class groups, is more efficient than the previous scheme and still provably secure without assuming the random oracle model.

We recall the idea of authentication "flat tree" [CD95,CD96,Büt99] and put it on top of our signature scheme.

4.1 Description of the Scheme

Let l and d be new integer parameters. We will construct a tree of degree l and depth d.

As in the original protocol we choose an abelian group G of a random order n according to the distribution D.

We generate a list of l primes $e_1, \ldots, e_l$ such that $B + C < e_i < 2\,(B + C)$ for each $1 \leq i \leq l$.

Let us consider the root of the authentication tree. As in the simulation in the security proof of the original protocol we choose random $z_0, \omega_0 \in G$ and a prime e' satisfying $B + C < e' < 2\,(B + C)$ and $e' \neq e_i$ for all $1 \leq i \leq l$. We compute $h_0 = z_0^{\prod_{1 \leq i \leq l} e_i}$ and $x_0 = \omega_0^{\prod_{1 \leq i \leq l} e_i}$.

The values h_0, x_0 and e' are publicly known.

For each child i of the root we choose random elements $z_{1,i}, \omega_{1,i} \in G$ and compute $h_{1,i} = z_{1,i}^{\prod_{1 \leq j \leq l} e_j}$ and $x_{1,i} = \omega_{1,i}^{\prod_{1 \leq j \leq l} e_j}$. Now as in the simulation we can sign the pair $h_{1,i}, x_{1,i}$ using h_0, x_0 and the exponent e_i. Let $\sigma_{1,i}$ be that signature.

If we repeat that authentication procedure for each node using its signed values h and x together with exponent e', we can construct a tree of degree l and depth d. We will have l^d leaves. Instead of constructing new values h and x for each leaf we will sign messages using leaves. Hence we will be able to sign l^d messages (one message for each leaf).

It is important to note that we don't need to remember the whole tree in order to sign messages. If we use leaves in order "from the left", the path from the root to the leaf is sufficient to construct a signature of a new message (for details see [CD95,CD96,Büt99]). Let the nodes on the path be (h_0, x_0, h_1, x_1 with signature $\sigma_1, \dots, h_{d-1}, x_{d-1}$ with signature σ_{d-1}). Using the values h_{d-1}, x_{d-1}, the public exponent e' and appropriate exponent e_i we sign a message m. That gives us the signature σ_d. Finally the signature of message m is the list $\sigma = (h_1, x_1, \sigma_1, \dots, h_{d-1}, x_{d-1}, \sigma_{d-1}, \sigma_d)$.

To verify the signature σ we verify partial signatures $\sigma_1, \dots, \sigma_d$ consecutively using the original scheme with appropriate values of x, h and e_i.

4.2 Security Proof.

Theorem 3. *Let l^d be polynomial in the security parameter k. Assume that the signature scheme from previous scheme was unforgeable under adaptively chosen message attack. Then our tree authentication scheme is unforgeable under adaptive chosen message attack.*

Proof. We can see our scheme as calling $\left(1 + l + \cdots + l^{d-1}\right)$ instances of the original scheme from the previous section with the same values e' and $e_1, \dots, e_l$.

Suppose that the adversary A is able to forge our signature scheme with authentication tree with non-negligible probability. We construct a simulation which will allow us to break the original scheme.

Let S be an original scheme with values x, h, e' and $e_1, \dots, e_l$. We construct our authentication tree associating one random internal node with S.

First we answer at most l^d forger's requests to sign messages using the simulated "flat tree" signature scheme. Whenever we need a partial signature from the node associated with S we ask S for that signature (it happens at most l times).

Suppose that the forger is able to sign a new message m. His valid signature σ determines a path $h'_1, x'_1, \sigma'_1, \dots, h'_{d-1}, x'_{d-1}, \sigma'_{d-1}, \sigma'_d$. Since the partial signatures $\sigma'_1, \dots, \sigma'_d$ contain the exponents $e'_1, \dots, e'_d$ respectively, we know the path $h_1, x_1, \sigma_1, \dots, h_t, x_t$ with $t \leq d-1$ which was copied by the forger (if $t < d-1$ then $\left(\sigma'_t, h'_{t+1}, x'_{t+1}\right) \neq (\sigma_t, h_{t+1}, x_{t+1})$).

The adversary A has forged a partial signature using the instance of original scheme associated with the node h_t, x_t. Since the instances constructed by us are indistinguishable from the given instance S, the adversary A has forged the signature scheme S with non-negligible probability.

4.3 Efficiency Analysis

Let $k = \log(B + C)$.

Each signature contains d partial signatures. Hence the size of the signature is $O(dk)$.

The signer has to keep the list of l primes, which corresponds to $O(kl)$ bits and a path of size $O(dk)$ bits.

For signing a new signature the signer has to move to a new path. It requires computing at most d partial signatures, on average less than $1 + 2/\ell$. Computing a new partial signature in a straightforward way requires $O(lk)$ group operations (generally computing roots takes this time).

It follows from Fact 4 that we could have done some precomputation for each node requiring $O(kl \log \ell)$ multiplications in G. That would give us on average $O(k \log \ell)$ group operations for each partial signature. Then it would be sufficient to use $O(k)$ group multiplications for each partial signature. However this solution requires keeping $O(kl)$ bits in memory for each node. If we store the results of precomputation for each node on the path, it will occupy $O(kld)$ bits in the memory of signer.

Note that one could also apply the so called pebbling algorithm by Itkis and Reyzin [IR01], which would result in faster precomputation ($O(kI)$ group multiplications per node) and shorter keys ($O(k \log I)$ bits per node), but slower signing ($O(k \log I)$ group multiplications per partial signature).

The verifier just checks d partial signatures (see previous section for details).

4.4 Realistic Parameters

Let $l = 1000$ and $d = 3$. Then we are able to sign one billion messages, which should be sufficient in real life applications. Signing messages will be still fast and the signer's system will require a reasonable amount of data to be stored.

References

[BBHM00] Ingrid Biehl, Johannes Buchmann, Safuat Hamdy, and Andreas Meyer. A signature scheme based on the intractability of computing roots. Technical Report 1/00, Darmstadt University of Technology, 2000.

[BL96] D. Boneh and R. J. Lipton. Algorithms for black-box fields and their application to cryptography. *Lecture Notes in Computer Science*, 1109:283–297, 1996.

[Bue84] Duncan A. Buell. The expectation of success using a Monte Carlo factoring method—some statistics on quadratic class numbers. *Mathematics of Computation*, 43(167):313–327, July 1984.

[Büt99] Marc Bütikofer. An abstraction of the Cramer-Damgård signature scheme based on tribes of q-one-way-group-homomorphisms. ETH Zürich, 1999.

[BV98] D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. In *Proceedings of Eurocrypt '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 59–71, 1998.

[BW88] Johannes Buchmann and H. C. Williams. A key-exchange system based on imaginary quadratic fields. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 1(2):107–118, 1988.

[CD95] R. Cramer and I. Damgård. Secure signature schemes based on interactive protocols. In *Proceedings of CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 297–310, 1995.

[CD96] R. Cramer and I. Damgård. New generation of secure and practical RSA-Based signatures. In *Advances in Cryptology — CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 173–185. Springer Verlag, 1996.

[CHL84] H. Cohen and Jr. H.W. Lenstra. Heuristics on class groups of number fields. In *Number Theory, Noordvijkerhout 1983*, volume 1068 of *Lecture Notes in Math.*, pages 33–62, 1984.

[CS99] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. In *ACM Conference on Computer and Communications Security*, pages 46–51, 1999.

[DF02] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Y. Zheng, editor, *Advances in Cryptology'ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, 2002.

[Fis00] Marc Fischlin. A note on security proofs in the generic model. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT ' 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 458–469, Kyoto, Japan, 2000. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany.

[HM00] Safuat Hamdy and Bodo Möller. Security of cryptosystems based on class groups of imaginary quadratic orders. In T. Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 234–247. Springer-Verlag, 2000.

[HMT99] Detlef Hühnlein, Andreas Meyer, and Tsuyoshi Takagi. Rabin and RSA analogues based on non-maximal imaginary quadratic orders. Technical Report 7/99, Darmstadt University of Technology, 1999.

[HT99] Detlef Hühnlein and Tsuyoshi Takagi. Reducing logarithms in totally non-maximal imaginary quadratic orders to logarithms in finite fields. In *Proceedings of ASIACRYPT '99*, volume 1716 of *LNCS*, pages 219–231, 1999.

[IR01] Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In J. Kilian, editor, *CRYPTO*, volume 2139 of *LNCS*, pages 332–354. Springer Verlag, 2001.

[Jac99] M. Jacobson. *Subexponential class group computation in quadratic orders.* PhD thesis, Technische Universitat Darmstadt, Darmstadt, Germany, 1999.

[Lag80] J. C. Lagarias. Worst-case complexity bounds for algorithms in the theory of integral quadratic forms. *J. Algorithms*, 1(2):142–186, 1980.

[MKI88] Tsutomu Matsumoto, Koki Kato, and Hideki Imai. Speeding up secret computations with insecure auxiliary devices. In S. Goldwasser, editor, *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 497–506. Springer-Verlag, 1990, 21–25 August 1988.

[Mor93] Pieter Moree. *On the psixyology of Diophantine equations.* PhD thesis, Leiden University, 1993.

[MW98a] U. Maurer and S. Wolf. Lower bounds on generic algorithms in groups. *Lecture Notes in Computer Science*, 1403:72–84, 1998.

[MW98b] J. Merkle and R. Werchner. On the security of server-aided RSA protocols. *Lecture Notes in Computer Science*, 1431:99–116, 1998.

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994. Translated from Matematicheskie Zametki, 55(2):91–101, 1994.

[NS01] P. Q. Nguyen and I.E. Shparlinski. On the insecurity of a server-aided RSA protocol. In C. Boyd, editor, *Advances in Cryptology—Asiacrypt'2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 21–35. Springer-Verlag, 2001.

[Sch91] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 4(3):161–174, 1991.

[Sch02] Claus Peter Schnorr. Security of DL-encryption and signatures against generic attacks - a survey. In K.Alster, H.C.Williams, and J.Urbanowicz, editors, *Proceedings of Public-Key Cryptography and Computational Number Theory Conference, Warsaw, September, 2000.* Walter De Gruyter, 2002.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology: Eurocrypt '97*, pages 256–266, 1997.

[SJ99] Claus Peter Schnorr and Markus Jakobsson. Security of discrete log cryptosystems in the random oracle + generic model. In *Conference on The Mathematics of Public-Key Cryptography*, The Fields Institute, Toronto, Canada, 1999.

[SJ00] Claus Peter Schnorr and Markus Jakobsson. Security of signed ElGamal encryption. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT ' 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 73–89, Kyoto, Japan, 2000. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany.