A Statistically-Hiding Integer Commitment Scheme based on Groups with Hidden Order

Ivan Damgård[1] and Eiichiro Fujisaki[2]

[1] BRICS, Dept. of Computer Science, Aarhus University, Ny Munkegade AARHUS C DK-8000 Denmark
E-mail: ivan@daimi.au.dk
[2] NTT Labs, 1-1 Hikarinooka, Yokosuka-shi 239-0847 Japan.
E-mail: fujisaki@isl.ntt.co.jp.

Abstract. We present a statistically-hiding commitment scheme allowing commitment to arbitrary size integers, based on any (Abelian) group with certain properties, most importantly, that it is hard for the committer to compute its order. We also give efficient zero-knowledge protocols for proving knowledge of the contents of commitments and for verifying multiplicative relations over the integers on committed values. The scheme can be seen as a generalization, with a slight modification, of the earlier scheme of Fujisaki and Okamoto [14]. The reasons we revisit the earlier scheme and give some modification to it are as follows:

- The earlier scheme [14] has some gaps in the proof of soundness of the associated protocols, one of which presents a non-trivial problem which, to the best of our knowledge, has remained open until now. We fill all the gaps here using additional ideas including minor modification of the form of a commitment.
- Although related works such as [8, 3, 10, 4] do not suffer from the main problem we solve here, the reason for this is that they use "commitments" with a single base (i.e., of form $c = g^s \bmod n$). Such commitments, however, cannot satisfy the standard hiding property for commitments, and hence protocols using them cannot in general be (honest-verifier) zero-knowledge nor witness indistinguishable.
- In a computationally convincing proof of knowledge where the prover produces the common input (which is the type of protocol we look at here), one cannot completely exclude the possibility that a prover manages to produce a common input on which he can cheat easily. This means that the standard definition of proofs of knowledge cannot be satisfied. Therefore we introduce a new definition for computationally convincing proofs of knowledge, designed to handle the case where the common input is chosen by the (possibly cheating) prover.
- Our results apply to any group with suitable properties. In particular, they apply to a much larger class of RSA moduli than the safe prime products proposed in [14] — Potential examples include RSA moduli, class groups and, with a slight modification, even non-Abelian groups.

Our scheme can replace the earlier one in various other constructions, such as the efficient interval proofs of Boudot [4] and the efficient proofs for the product of two safe primes proposed by Camenisch and Michels [9].

1 Introduction

1.1 Statistically-hiding commitment and associated protocols

The notion of commitment is at the heart of many cryptographic protocols. The basic functionality one wants from a commitment is that the committer may choose in private a secret s from some set S and release some information, *the commitment* to a verifier, such that: even though the scheme is *hiding*, i.e., the verifier cannot compute anything about s from the commitment, it is also *binding*, i.e., the committer cannot change his mind after having committed, but he can later open the commitment to reveal s, and convince the verifier that this was indeed the original value committed to.

In many applications, one wants extra functionality from a commitment scheme, for instance that the committer can prove in zero-knowledge that he knows how to open a given commitment, in particular that he knows the value committed to. Also, if S has an algebraic structure, say as a ring or a group, it can be very useful to have a *multiplication protocol*, i.e., a zero-knowledge protocol in which the committer can prove that committed values a, b, c satisfy $ab = c$. If S is a ring, one can often, in addition, achieve that from commitments to $a, b \in S$, the verifier can compute a commitment to $a + b$ without interacting with the committer.

One example of such a scheme where $S = \mathbb{Z}/q\mathbb{Z}$, where q is a prime, is the scheme of Pedersen [17]. For the associated protocols and additional examples, see [7]. In the vast majority of examples known, the set S is $\mathbb{Z}/m\mathbb{Z}$ for some m, where m may or may not be a prime. A multiplication protocol for such a scheme is a protocol by which one can demonstrate that for committed numbers a, b, c, $ab = c \bmod m$ holds. However, there are several important cases where what you actually need is something stronger, namely to be able to prove that $ab = c$ holds *over the integers*. One example of this is if you want to show that a committed number s is an RSA signature on a given message a w.r.t. public key $n, 3$. What we want to know is that $a = s^3 + tn$ for some t, and this of course must be true over the integers and not just modulo m. Of course, one might be able to solve this by choosing the commitment scheme such that $m = n$, but this requires that at least you know n at the time the commitment scheme was set up, and also a new instance of the commitment scheme for each n. This is often unreasonable in practice. There are other ways around the problem, see for instance [12], but the protocols are far from optimal, typically one has to resort to "binary cut-and-choose", which means communication complexity at least quadratic in the security parameter. Another example of the need for relations over the integers is the efficient zero-knowledge proofs of Boudot [4] for demonstrating that a committed number is in a given interval. Here, it is crucial for efficiency that one can prove efficiently that committed numbers a, b satisfy $b = a^2$ over the integers.

It should be clear that what we really need here is an *integer* commitment scheme, that is, a scheme where $S = \mathbb{Z}$ (or at least some large finite interval), and where there is an efficient multiplication protocol that works over the integers.

Here, by efficient, we mean constant round protocols requiring only communication linear in the security parameter.

1.2 The earlier scheme with statistically-hiding commitment

In [14], Okamoto and the second author of this paper presented the first efficient integer commitment scheme and also suggested an efficient multiplication protocol. The scheme is based on the strong RSA assumption suggested in [2, 14]. However, via private communication, we found some gaps in the proof of soundness of the associated protocols, one of which we think presents a non-trivial problem which, to the best of our knowledge, has remained open until now. Later in the paper we give a short explanation of the problem in the proof from [14]. We fill all the gaps here using additional idea including a minor modification of the form of a commitment.

1.3 Other related works

There are several related works inspired by [14] such as [8, 3, 10, 4]. The protocols constructed there generally do not suffer from the main problem we mentioned above. However, the reason for this is that they use "commitments" with a single base, i.e., a commitment to s is of form $c = g^s \bmod n$. Such a commitment does not satisfy the standard hiding property for commitments. For instance, if a prover commits twice to the same value, this is immediately visible. Thus derived protocols using such commitments are not in general (honest-verifier) zero-knowledge nor witness indistinguishable.

Boudot [5] pointed out another problem in the proof of soundness — In this type of protocols (based on a group with a hidden order), the natural protocol for showing that one knows how to open a commitment c can in fact only show that the prover can open c or $-c$ (a problem that even [8, 3, 10, 4] cannot avoid). This is not so serious in practice but we suggest a solution to this problem too, by changing the way in which commitments are opened.

1.4 Our scheme

In this paper, we present a commitment scheme that may be seen as a generalization of the Fujisaki-Okamoto scheme. We start with an arbitrary Abelian group G, with some basic properties. We assume that the verifier can choose the group and publish a *description* of it that allows anyone to compute the group and inversion operations in G. For the RSA case, this amounts to publishing the modulus n. The most important extra property we need is that it is hard, given the description, to extract the roots of a given random element in G. This is just a natural generalization of the strong RSA assumption. Some extra technical conditions are needed as well, we detail those later. We then build from this an integer commitment scheme, as well as a zero-knowledge protocol for proving knowledge of how to open a commitment, and an efficient zero-knowledge multiplication protocol. In order to analyze these protocols, we introduce a new definition of computationally convincing proofs of knowledge, designed to handle the case where the common input is chosen by the (possibly cheating) prover. Our analysis is done in the exact security setting.

If we specialize to the case where $G = (\mathbb{Z}/n\mathbb{Z})^\times$ for an RSA modulus n, we obtain - modulo some technical changes - the commitment scheme of Fujisaki and Okamoto, in particular we get what appears to be the first secure multiplication protocol for this type of scheme. In addition, the conditions we need on G turn out to translate into conditions on n that are much milder than those needed in the original paper [14], namely that $n = pq$ is a safe prime product. We only need that $\gcd(p-1, q-1) = 2$ and $p-1, q-1$ don't have too many small prime factors (whose precise description follows below). Finally, our construction is applicable to groups other than RSA, for instance class groups. Here, it should be noted that finding roots in a class group seems to require finding the order of the group, and this problem is known to be at least as hard as factoring, and may in fact be harder.

Our commitment scheme and protocols are not exactly the same as those of [14], even when specialized to $G = (\mathbb{Z}/n\mathbb{Z})^\times$. However, with some minor technical changes of the commitment scheme in [14], one can give correct proofs for soundness of their protocols following the ideas we give here. However, our protocols are slightly more efficient than those of [14].

There are several variants for our protocols that we do not explain here due to space limitations, except for a few extensions given in Appendix B.

2 Model

As usual, probability $\epsilon(k)$ will be called *negligible* if for all polynomials $f(\cdot)$, we have $\epsilon(k) \leq 1/f(k)$ for all large enough k. On the other hand, $1 - \epsilon(k)$ will be called *overwhelming* if $\epsilon(k)$ is negligible. Also, we say $\epsilon(k)$ is *significant* if for some polynomial $f(k)$, we have $\epsilon(k) \geq 1/f(k)$ for all large enough k.

Suppose now that we are given a probabilistic polynomial time algorithm $\mathcal{G}$ which on input 1^k outputs a description $\mathsf{descr}(G)$ of a finite Abelian group G, where we assume one can efficiently verify from $\mathsf{descr}(G)$ that it actually specifies such a group. The algorithm may also output some side information, such as the order of G, or the prime factorization of the order; it may even be possible to ensure that the order of the group satisfies certain conditions. An example of such a $\mathcal{G}$ is an RSA key generation algorithm — in this case it is indeed possible to generate a group with known and controlled order.

Given $\mathsf{descr}(G)$, we assume that one can compute efficiently some estimates on the order, $2^A \leq \mathsf{ord}(G) \leq 2^B$, where A and B are polynomial in k. We also assume that elements can be sampled randomly from the group and that inversion and group operation can be computed efficiently.

In order for our protocols to work, we need, loosely speaking, that it is hard to find non-trivial roots of elements in G. Furthermore, we need a condition on the structure of G's output by $\mathcal{G}$. Loosely speaking, we need that G has a large subgroup with only large prime factors in its order. To make this more precise, we assume that two functions are associated with $\mathcal{G}$: $C(\cdot)$, $l(\cdot)$, that map positive integers to positive integers. Typically, $C(k)$ is super-polynomially large as a function of k, whereas $l(k)$ is always a polynomial. For any G produced by $\mathcal{G}$ on input 1^k, we will consider primes greater that $C(k)$ as being "large". By the structure theorem for Abelian groups, we can always write $G = U \times H$, where the order of H has only prime factors larger than $C(k)$, and the order of U has only primes factors at most $C(k)$. We say $|H|$ is $C(k)$-rough, as opposed to being $C(k)$-smooth, which means a number has only prime factors less than $C(k)$. Thus $l_G := |U|$ is $C(k)$-smooth.

We are now ready to state our assumptions about groups output by $\mathcal{G}$:

Group assumption. For any G generated by $\mathcal{G}$ on input 1^k the following hold:

1. Write $G = U \times H$ as above, with $C(k)$-smooth, $l_G = |U|$ and $C(k)$-rough $|H|$. Then $l_G \leq l(k)$ and $\mathsf{descr}(G)$ includes l_G.
2. For any string Y, when given $\mathsf{descr}(G)$, it can be decided in (deterministic) polynomial-time in k whether Y represents an element in G.

Root assumption. Let A be a probabilistic algorithm. We run $\mathcal{G}$ on input 1^k to get $\mathsf{descr}(G)$. We give $\mathsf{descr}(G)$ and a random $Y \in G$ as an input to A. We say that A solves the root problem if A outputs an integer $e(> 1)$, $X \in G$, and $\mu \in U$ such that $Y = \mu X^e$ (where $\mu \in U$ can be verified by checking that $\mu^{l_G} = 1 \in G$). In particular, we say that the root problem is $(t(k), \epsilon(k))$-secure if for k, any adversary A that runs in time at most $t(k)$, solves the root problem with probability of at most $\epsilon(k)$. The probability is taken over the coin tosses of $\mathcal{G}$ and A, as well as the random choice in G.

Some remarks on the assumptions:

The condition that $l_G \leq l(k)$ says that G has many elements with only large prime factors in their orders: If Y is chosen randomly in G, then there is a significant probability, $1/l(k)$, that the order of Y is $C(k)$-rough. We want to stress that it is essentially important that membership in G can be decided efficiently – although this property is often ignored and forgotten, this was one reason why proofs of soundness for the earlier protocols suggested in [14] were incomplete. We will assume throughout that when a party receives an element that is supposed to be in G, membership in G is always checked.

The assumption that l_G is known and is part of the description can be removed, if in the protocols to follow, one replaces exponentiations to the power l_G by exponentiations to $l(k)!$. The price is loss of efficiency, since $l(k)! \sim \sqrt{2\pi} l(k)^{l(k)+1/2} e^{-l(k)} >> l_G$. However, the cost to exponentiate to power $l(k)!$ is still polynomial in k.

The second assumption is a generalization of the strong RSA assumption – we require that extracting non-trivial roots is hard, even if one is allowed to multiply the input by an element of relatively small known order. We may think of this as root extraction in a factor group: when the adversary algorithm gets an input element Y, this represents an element $\bar{Y}$ in the quotient group G/U, and the adversary's task actually is to extract a non-trivial (e'th) root of $\bar{Y}$ in

G/U. He must, however, demonstrate that his answer when raised to the e'th power represents the same element as does Y. We require he does this by also producing μ.

If we specialize to the RSA case, i.e., $G = (\mathbb{Z}/n\mathbb{Z})^{\times}$ for an RSA modulus n, it may seem that the root assumption as we defined it here would make an even stronger requirement than the standard strong RSA assumption [2, 14] – since the adversary in our case is given l_G and does not have to find a root of Y, but of μY for any $\mu \in U$. This is not the case, however. We now show that RSA moduli can be constructed such that our assumptions are satisfied for these groups, based only on the strong RSA assumption in its standard form. Suppose we make a k-bit modulus $n = pq$ such that $\gcd(p-1, q-1) = 2$. We choose $C(k)$ as some super-polynomial function much less than 2^k, for instance $C(k) = 2^{k/10}$, and we set $l(k) = k$. We construct p, q such that the factor of $(p-1)(q-1)$ with prime factors less than $C(k)$ is in $O(k)$ (this factor is l_G, where $G = (\mathbb{Z}/n\mathbb{Z})^{\times}$). We then set $G = (\mathbb{Z}/n\mathbb{Z})^{\times}$ and $\mathsf{descr}(G) = \{n, l_G\}$. Now, the root assumption (in its asymptotic form) turns out to be equivalent to the standard strong RSA assumption. First note that it makes little difference whether l_G is known since it can be guessed with significant probability. Then suppose algorithm A on input Y, n, l_G finds X, e, μ such that $Y = \mu X^e, \mu^{l_G} = 1$. Now, if there is non-negligible probability that $\mu \neq \pm 1$, we can use μ, l_G to factor n, namely we first factor l_G and then we can find an element $\tilde{\mu}$ of known prime order s. If $s = 2$, $\tilde{\mu}$ is a non-trivial square root of 1 and $\gcd(\tilde{\mu} - 1, n)$ is a factor in n. But if s is odd, it cannot divide both $p - 1$ and $q - 1$ and therefore $\tilde{\mu}$ must be congruent to 1 modulo one of p or q and be different from 1 modulo the other. Hence, also in this case, $\gcd(\tilde{\mu} - 1, n)$ is a non-trivial factor of n. On the other hand, if $\mu = \pm 1$ with non-negligible probability, we can solve the strong RSA problem: given input $h \in (\mathbb{Z}/n\mathbb{Z})^{\times}$, we choose a random bit b and give $(-1)^b h$ as input to A. Since A receives the same input distribution as usual, it outputs a non-trivial root of $(-1)^b Y$ or $-(-1)^b Y$ with good probability. Since A's choice of the sign cannot be correlated to our choice of b, we obtain a root of Y with non-negligible probability.

Note that a special case of this construction of n is when $n = pq$ is a safe prime product, i.e., $(p-1)/2, (q-1)/2$ are primes, but evidently the construction covers a much larger class of moduli.

3 Some definitions

We will often use the concepts of zero-knowledge and computational/statistical indistinguishability. For definitions of these, refer to [16]. The definitions below are all in the exact security style. It is straightforward to derive asymptotic type definitions from the exact-security style ones.

We then define the type of commitment scheme we will look at. Our commitments will be statistically (unconditionally) hiding and computationally binding. Concretely, a commitment scheme consists of a probabilistic polynomial time *key generator* $\mathcal{H}$, which on input 1^k outputs a public key pk and a witness w.

We let L_H be the set of public keys that $\mathcal{H}$ can produce as an output, w is an NP-witness to the fact that $pk \in L_H$. The scheme also defines an algorithm commit that takes as inputs pk, a string s to be committed to, and a random string r, both of lengths that are fixed polynomials in k. The output is a commitment to s, $\mathsf{commit}_{pk}(s, r)$ and a string u. Finally we have an algorithm verify that takes inputs pk, commitment c, and strings s, u, where the output (denoted $\mathsf{verify}_{pk}(c, s, u)$) is *accept* or $reject$.

Such a scheme can be used by a committer C and a receiver R as follows: in the *set-up* phase, R runs $\mathcal{H}$ to get pk, w, sends pk and uses w to give a zero-knowledge interactive proof [16] that $pk \in L_H$. C can commit to s by running *commit* on pk, s and random input r, sending $c = \mathsf{commit}_{pk}(s, r)$ to R and keeping r secret. Opening takes place by revealing s, r to R, who can then check that $\mathsf{verify}_{pk}(c, s, r) = accept$.

We then require the following:

Hiding: For $pk \in L_H$, uniform r, r' and any s, s', we have that the distributions of $\mathsf{commit}_{pk}(s, r)$ and $\mathsf{commit}_{pk}(s', r')$ are (statistically) indistinguishable (as defined in [16]).

Binding: We say that binding is $(t(k), \epsilon(k))$-secure if it holds that for any C running in time at most $t(k)$, the probability that C on input pk computes s, r, s', r' such that $\mathsf{commit}_{pk}(s, r) = \mathsf{commit}_{pk}(s', r')$ and $s \neq s'$, is at most $\epsilon(k)$.

We will also need to consider proofs of knowledge in the following. For this, we use a modification of the definition of Bellare and Goldreich, in the version for computationally convincing proofs of knowledge [6]. Let a binary relation R be given, where a prover P and a verifier V are both probabilistic polynomial time interactive Turing machines. Intuitively, the prover's claim is that for a given common input c, he knows w such that $(c, w) \in R$.

To define this in our setting, we cannot use the original definition in [6] without change. This is because it asks that the soundness of the protocol holds for *all* (large enough) instances c. In our scheme, this is more than we can reasonably ask for: in our case, one may think of c as a commitment and w as the string P can use to open c. Furthermore, the scenario is that P sees the public key of the scheme, produces the commitment and then tries to prove he knows how to open it. But this proof is only computationally convincing in our case, so a cheating prover may have some chance of producing, based on the public key, a commitment he cannot open, but where the proof nevertheless is successful with large probability. This can typically happen if the prover manages to compute some trapdoor information associated with the public key. This information can always be guessed with non-zero probability and so the problem cannot be completely avoided, but we can at least require that it occurs with only small probability. In our definition of soundness, therefore, we first let the prover see a public piece of information, he then produces c and conducts the proof. A cheating prover P^* wins if the standard soundness requirement fails for this c, and we are satisfied with the proof system if within some time bound P^* can only win with some bounded (small) probability.

For the above definition we need to consider a *relation generator*, algorithm $\mathcal{R}$, that takes 1^k as an input and produces as an output a description of a binary relation R. By this we mean a string containing information is sufficient to sample efficiently random pairs $(c, w) \in R$ and to test membership in R efficiently. We will use R to denote both this description and the relation itself. For instance, $\mathcal{R}$ might be the key generator for a commitment scheme, and we can think of the public key pk as defining a relation consisting of pairs $(c, (s, r))$ for which $\mathsf{verify}_{pk}(c, s, r) = accept$.

A prover in our setting is a machine P who gets R as an input, outputs a string c and finally conducts the interactive proof with a verifier V using R, c as common input. For convenience, we want to be able to refer to P's strategy when executing the proof as a separate machine. Therefore, from P we define a machine P_{view} which starts in the state P is in after having seen view view and having produced c. P_{view} then conducts the protocol with V following P's algorithm. The view view contains all inputs, messages exchanged and random coins so in particular c is determined by view. Note that the distribution of view is taken over the random coins of both P and $\mathcal{R}$. We let $\epsilon_{\mathsf{view},P}$ be the probability with which P_{view} makes V accept, i.e. $\epsilon_{\mathsf{view},P}$ is P's probability to make V accept, conditioned on view.

An *extractor* will be a machine M that gets R, c as an input, has black-box access to P_{view} for some view consistent with c. and computes a witness w such that $(c, w) \in R$. The intuition is that the prover "knows" w if we can use M to extract it from him. To measure this, we need a *knowledge error* function $\kappa()$. Intuitively, $\kappa(k)$ is the probability that the prover can cheat on input generated from security parameter value k, i.e., make V accept while knowing nothing about w.

Definition 1. *For some given cheating prover P^*, extractor M and polynomial $p()$, we say M* fails *on view* view *if $\epsilon_{\mathsf{view},P^*} > \kappa(k)$, if the expected running time of M using P^*_{view} as oracle, is greater than $\frac{p(k)}{\epsilon_{\mathsf{view},P^*} - \kappa(k)}$.*

This definition is motivated by the fact that the standard knowledge soundness requirement from [6] says that the extractor must run in expected time *at most* the bound in the definition. Note that in any situation where P^* has produced c having seen view, it is well defined whether M fails or not. Intuitively, one may think of this as saying that if M does not fail in a given situation, then P^* really "must know" a witness for c, in order to make V accept with a probability better than $\kappa(k)$.

Definition 2. *Let $\mathcal{R}$ be a probabilistic polynomial time relation generator, and let a protocol (P, V), a knowledge extractor M, polynomial $p()$ and knowledge error function $\kappa()$ be given. Consider the following experiment with input k: $R := \mathcal{R}(1^k), c := P^*(R)$ (this defines view* view*). We define the* advantage *of P^*,* $\mathsf{Adv}_{\kappa,M,p}(P^*, k)$ *as the probability that M fails on the view generated by this experiment. This probability is taken over the random coins of $\mathcal{R}$, P^*.*

Finally, for a relation R, we let, as usual, $L_R = \{c | \ \exists w : (c, w) \in R\}$. We are now ready to define computationally convincing proofs of knowledge:

Definition 3. *Let $\mathcal{R}$ be a probabilistic polynomial time relation generator. We say that (P, V) is a computationally convincing proof of knowledge for $\mathcal{R}$, with knowledge error $\kappa()$, failure probability $\nu()$ and time bound $t()$, if the following hold:*

knowledge completeness *The honest prover P receives $R \leftarrow \mathcal{R}(1^k)$, produces $(c, w) \in R$, sends c to V and finally conducts the protocol with V, who accepts with overwhelming probability in k.*

knowledge soundness *There exists a polynomial $p()$ and an extractor M, such that for all provers P^* running in time at most $t(k)$, $\mathsf{Adv}_{\kappa,M,p}(P^*, k) \leq \nu(k)$.*

4 The Commitment Scheme

Based on the above model, the goal is to make a commitment scheme with protocols to verify various claims on committed values. The basic scheme is that the verifier V (the receiver of commitments) will run $\mathcal{G}$ and send $\mathsf{descr}(G)$ (and more information to be described later) to the prover P (the committer).

Set-up V runs $\mathcal{G}(1^k)$ and chooses a random element $h \in G$, such that $\mathsf{ord}(h)$ is $C(k)$-rough (this can be done by raising a random element to power l_G). Now V sets $g = h^\alpha$, where α is randomly chosen in $[0..2^{2B+k}]$. V sends $\mathsf{descr}(G), g, h$ to P and proves that $g \in < h >$, by the standard zero-knowledge discrete log protocol with binary challenges: in one iteration of this, V sends $a = h^R$ for a random $R \in [0..2^{2B+2k}]$. P selects a random bit b, and V replies with $z = R + b\alpha$. P checks that $h^z = ag^b$. Repeating this k times results in a soundness error of 2^{-k}, and the protocol is easily seen to be statistical zero-knowledge. This is not a very efficient solution, but it only needs to be done once and only in the set-up phase.

Commit To commit to an integer x, P chooses r at random in $[0..2^{B+k}]$, sends $c = g^x h^r$ to V, and stores x, r for later use.

Open To open a commitment, P must send x, r, μ such that $c = \mu g^x h^r$ and $\mu^{l_G} = 1$. An honest prover can always use $\mu = 1$. Although this gives a dishonest prover extra freedom, this in no way makes the commitment scheme weaker: the binding property still holds, as we argue below. Indeed, recalling our comments on the root assumption, one may think of the scheme as taking place in the quotient group G/U where U is the subgroup in G consisting of all elements of C-smooth order. From this point of view, the opening condition simply ensures that the prover opens something representing the same element as c in G/U (The quotient group is defined canonically so that $c \equiv \bar{c} \pmod U$ iff there is a $\mu \in U$ such that $\bar{c} = \mu c \in G$).

As for hiding, note that P verifies initially that $g \in < h >$. Hence, since r is chosen with bit length at least $k + \log_2(\mathsf{ord}(h))$, c is statistically close to uniform in $< h >$, for any value of x.

As for binding, we consider any prover P^* who can create c and the corresponding valid distinct openings, (μ, x, r) and (μ', x', r'). It follows that we get

$\mu g^x h^r = c = \mu' g^{x'} h^{r'}$. Recall that V creates g as $g = h^\alpha$. Plugging this in and raising both sides of the equation to l_G, we get that $h^{l_G(\alpha(x-x')+(r-r'))} = 1$. We can write $\alpha = q \cdot \mathsf{ord}(h) + res$ for integers q, res with $0 \leq res < \mathsf{ord}(h)$. Then from P^*'s point of view, res is uniquely determined from g, whereas there is an exponentially small amount of information on q (the only source of information is the proof that $g \in < h >$ which is statistical zero-knowledge). So P^*'s choice of x, x', r, r' is (almost) independent of q. It follows $l(\alpha(x-x') + (r-r')) = 0$ (as an integer) with probability exponentially small in k. Assuming this number is indeed non-zero, $\mathsf{ord}(h)$ must divide $M := \alpha(x-x') + (r-r')$ (since the order is $C(k)$-rough).

We can now use P^* to break the root assumption as follows: given input $\mathsf{descr}(G), h \in G$, we choose g as V would have done it, send $\mathsf{descr}(G), h, g$ to P^* and execute in the normal way the proof that $g \in < h >$. With probability of at least $1/l(k)$, h will have $C(k)$-rough order and everything has the same distribution as in a normal execution of the commitment scheme. Given that P^* breaks the binding property as described above, this allows us (except with negligible probability) to compute M, a multiple of the order of h. Now choose any t that is relatively prime to M and output $h^{t^{-1} \bmod M}$ and t.

Summarizing, we have:

Theorem 1. *Under the root assumption, the above scheme is an unconditionally hiding and computationally binding commitment scheme. If the root assumption is $(t(k), \epsilon(k))$-secure, then the binding property is $\left(t(k), \frac{\epsilon(k)l(k)}{1-2^{-\gamma k}}\right)$-secure for some constant γ.*

5 Associated Protocols

5.1 Proving you know how to open

The following protocol can be used by P to show that he can open a given commitment $c = g^x h^r$.

We will assume that x is in $[-T..T]$ where $T(> 0)$ is a public constant. T can be chosen arbitrarily large, and is only used to control the size of the prover's random choices, this allows an honest prover to ensure that the protocol hides the value of x, whenever $-T \leq x \leq T$. In any application of the scheme, one simply chooses T large enough to accommodate any choice of x an honest prover would need to make in the given scenario. The protocol guarantees an honest verifier that $-TC(k)(2^k+2) \leq x \leq TC(k)(2^k+2)$. To prove x is in some other (smaller) interval, other techniques exist, see e.g. [4].

1. P chooses $y \in [0..TC(k)2^k[, s \in [0..C(k)2^{B+2k}[$ at random and sends $d = g^y h^s$ to V.
2. V chooses at random $e \in [0..C(k)[$ and sends to P.
3. P sends $u = y + ex, v = s + er \in \mathbb{Z}$. V checks that $g^u h^v = dc^e$ and that $[-TC(k)..TC(k)(2^k+1)]$.

Define a relation generator $\mathcal{R}$ as follows: run $\mathcal{G}(1^k)$ to get G, choose $h \in H$ with $C(k)$-rough order, set $g = h^\alpha$ and output $\mathsf{descr}(G), g, h$. Then define the relation $R = \{(c, (\mu, x, r))|\ c, b \in G,\ c = \mu g^x h^r, \mu^{l_G} = 1, x \in [-TC(k)(2^k + 2)..TC(k)(2^k + 2)]\}$.

We now analyze to what extent the protocol above satisfies our definition of knowledge soundness, in particular for which knowledge error functions is the definition satisfied. Accordingly, let $\kappa()$ be any knowledge error function, such that $\kappa(k) \geq 4/C(k)$ for all k. We then must define an extractor M. Let a polynomial time prover P^* be given and let view be any view P^* may have after having produced a commitment c. Now, it can be shown that since there are $C(k)$ different challenges, then if $\epsilon_{\mathsf{view},P^*} > \kappa(k) \geq 4/C(k)$, standard rewinding techniques allow us to obtain in expected polynomial time a situation where, for a given d, P^* has correctly answered two different values e and e' with numbers u, v and u', v', so we get $g^{u-u'} h^{v-v'} = c^{e-e'}$. Let Rewind be a (probabilistic) procedure that creates e, e', u, v, u', v' in this way. A concrete algorithm for Rewind is given in Appendix A. It runs in expected time $56/\epsilon_{\mathsf{view},P^*}$, counting the time to do the protocol once with P^* as one step[3].

Assume without loss of generality that $e > e'$ and suppose that $(e - e')$ divides both $(u - u')$ and $(v - v')$. We now see that the element $\mu = g^{\frac{u-u'}{e-e'}} h^{\frac{v-v'}{e-e'}} c^{-1}$ satisfies that $\mu^{e-e'} = 1$. Since $e - e' < C(k)$, it follows that $\mathsf{ord}(\mu)$ is $C(k)$-smooth so that $\mu^{l_G} = 1$. So c can be correctly opened by sending $(u - u')/(e - e'), (v - v')/(e - e'), \mu$. Moreover, V's check on the size of u, u' implies that $(u - u')/(e - e')$ is in the required interval. A set of values e, e', u, u', v, v' is said to be *bad* if $e - e'$ does not divide both $u - u'$ and $v - v'$. The extractor M simply repeats calling Rewind (for this same c) until it gets a set of good values. We will analyze knowledge soundness with this M and the polynomial $p(k)$ from the definition set to the constant of 112. We start with a lemma that gives an exact bound on the security.

Lemma 1. *Let $\mathcal{R}$, (P, V), κ, M and $p()$ be as defined above. Given any prover P^*, there exists an algorithm $A(P^*)$ that solves the root problem defined by $\mathcal{G}(1^k)$ with probability $\frac{\mathsf{Adv}_{\kappa,M,p}(P^*,k)}{9l(k)}$ if $k \geq 6$, and runs in time $448 \cdot t_{P^*}(k)/\kappa(k)$ where $t_{P^*}(k)$ denotes the running time of P^* ($1/l(k)$ is an lower bound on the probability that a random element in G has $C(k)$-rough order).*

Proof. The algorithm claimed does the following: receive G, h as an input. Set $g = h^\alpha$ for random $\alpha \in [0..2^{2B+k}]$. We send g, h to the adversary, call Rewind and hope that we get get a set of bad values. However, we will only allow Rewind to do the protocol with the prover at most $448/\kappa(k)$ times. If Rewind runs longer than this, we abort it and stop. If we obtained a set of bad values, we attempt to compute a root of h as described below.

[3] Note that this is not completely trivial, as P^* is probabilistic: although its average success probability is $\epsilon_{\mathsf{view},P^*}$, it may not be equally successful for all choices of random coins. It is essential to get the claimed expected time that $\epsilon_{\mathsf{view},P^*} > 4/C(k)$, and not just $> 1/C(k)$

It is immediately clear that this algorithm has the claimed running time. We now look at the success probability. We will assume that h has $C(k)$-rough order. Since this happens with probability of at least $1/l(k)$, it is enough to show that the success probability under this assumption is at least the bound claimed times $l(k)$.

Note that the distribution of G, h, g that P^* receives here is exactly the same as in the real commitment scheme. Hence the probability of producing a view for which M fails, is exactly $\mathsf{Adv}_{\kappa,M,p}(P^*, k)$. Note also that given any view view where M fails, it must be the case that the values produced by Rewind are bad with probability of at least $1/2$. If this was not the case, then M could expect to find a way to open c after calling Rewind twice, which takes expected time $112/\epsilon_{view,P^*} \leq p(k)/(\epsilon_{view,P^*} - \kappa(k))$ so this would contradict the fact that M fails on view. So let E be the event that M fails on view and Rewind has returned a set of bad values. We now make the following

Claim: Given that E occurs, we can solve the root problem with probability of at least $1/2 - 2^{-k}$.

To see this, recall that Rewind returns e, e', u, u', v, v' such that $g^{u-u'}h^{v-v'} = c^{e-e'}$, and we have that $e - e'$ does not divide both $u - u'$ and $v - v'$. If we plug in that $g = h^\alpha$, we get

$$h^{\alpha(u-u')+v-v'} = c^{e-e'}$$

We then split in two cases:

Case 1: $e - e'$ **does not divide** $\alpha(u - u') + (v - v')$**.**

In this case, let $\beta = \gcd(e - e', \alpha(u - u') + (v - v'))$ (where by assumption $\beta < e - e' \leq C(k)$). Choose γ, δ such that

$$\gamma(e - e') + \delta(\alpha(u - u') + (v - v')) = \beta$$

We then get that

$$h^\beta = h^{\gamma(e-e')+\delta(\alpha(u-u')+(v-v'))} = (h^\gamma c^\delta)^{e-e'}.$$

If we set $\tilde{\mu} = (h^\gamma c^\delta)^{(e-e')/\beta}h^{-1}$, it is clear that $\tilde{\mu}^\beta = 1$, so since $\beta < C(k)$, $\mathsf{ord}(\tilde{\mu})$ is $C(k)$-smooth so that $\tilde{\mu}^{l_G} = 1$. Furthermore

$$h\tilde{\mu} = (h^\gamma c^\delta)^{(e-e')/\beta}$$

So in this case, we may output $h^\gamma c^\delta, (e - e')/\beta, \tilde{\mu}$, which is a solution to the root problem as we defined it earlier.

Case 2: $e - e'$ **divides** $\alpha(u - u') + (v - v')$**.**

Note that even in this case, we still have that $e-e'$ does not divide both $u-u'$ and $v - v'$. The goal will be to show that since the adversary does not know full information about our choice of α, this case happens with probability at most $(1/2 - 2^{-k})$, given that E occurs. Hence the previous case where we could solve the root problem happens with large probability, given E. Let q be some prime factor in $e - e'$ such that q^j is the maximal q-power dividing $e - e'$, and at least one of $u - u', v - v'$ are non-zero modulo q^j (such a q

must exist since $e - e'$ does not divide both of $u - u', v - v'$). Note that if q^j divides $u - u'$, it would have to divide $v - v'$ as well, which is a contradiction. So $u - u' \neq 0 \bmod q^j$. We can then write $\alpha = y + z \cdot \mathsf{ord}(h)$, where $y = \alpha \bmod \mathsf{ord}(h)$. Note that g represents all information the adversary has about α and y is uniquely determined from g, whereas z is completely unknown. Now, if indeed q^j divides $\alpha(u - u') + (v - v')$, we have

$$\alpha(u - u') + (v - v') = z(u - u')\mathsf{ord}(h) + y(u - u') + (v - v') = 0 \bmod q^j$$

Note that since $q < C(k)$ we have $\mathsf{ord}(h) \neq 0 \bmod q$. Now, from the adversary's point of view, z is chosen uniformly among at least 2^{B+k} values, and must satisfy the above equation in order for the bad case to occur. The number of solutions modulo q^j of this equation is at most $\gcd((u - u')\mathsf{ord}(h), q^j)$. This number is a power of q, but is at most q^{j-1}. Then, since 2^{B+k} is larger than q^j by a factor of at least 2^k, it follows that the distribution of $z \bmod q^j$ is statistically close to uniform in $\mathbb{Z}/q^j\mathbb{Z}$. In fact, the probability that z satisfies the equation is at most $1/q - 2^{-k} \leq 1/2 - 2^{-k}$. The claim above now follows.

Summarizing, we therefore have that for every view view where M fails, running Rewind will fail to solve the root problem with probability at most $1 - (1/2 - 2^{-k})/2 = 3/4 + 2^{-k-1}$. The expected number of executions of P^* needed to run rewind is at most $56/\epsilon_{view,P^*} \leq 56/\kappa(k)$. Thus Rewind is allowed to run for at least 8 times its expected running time, and so by the Markov rule it will run for longer with probability at most 1/8. Since the probability that view is bad in the first place is $\mathsf{Adv}_{\kappa,M,p}(P^*, k)$, the success probability of $A(P^*)$ is $\mathsf{Adv}_{\kappa,M,p}(P^*, k)(1 - 1/8 - 3/4 - 2^{-k-1}) \geq \mathsf{Adv}_{\kappa,M,p}(P^*, k)/9$ if $k \geq 6$. This finishes the proof.

Next we have:

Theorem 2. *If the root assumption is $(t'(k), \epsilon(k))$-secure, the above protocol is a computationally convincing proof of knowledge for $\mathcal{R}$ with knowledge error $\kappa(k)$, time bound $t(k)$ and failure probability $\nu(k)$, where $\nu(k) = 9\epsilon(k)l(k)$, $t(k) < t'(k)/448$ and $\kappa(k) = \max(4/C(k), 448t(k)/t'(k))$. If $-T \leq x \leq T$ (as it will be when the prover is honest), the protocol is honest verifier statistical zero-knowledge.*

Remark 1. There are a number of known techniques by which a protocol that is zero-knowledge in general can be constructed from an honest verifier zero-knowledge protocol.

Remark 2. Note that a prover playing against the commitment scheme as defined above will see both the public key pk and a zero-knowledge proof from V that pk was correctly chosen, whereas a prover in the proof of knowledge definition only sees the public key. This makes no difference, however, since the proof is statistical zero-knowledge and could always be simulated.

Proof. Completeness of this protocol is clear. It is honest verifier statistical zero-knowledge: to simulate we can choose at random $u \in [0..TC(k)2^k[$, $v \in [0..C(k)2^{B+2k}[$, $e \in [0..C(k)[$ and set $d = g^u h^v$. The rest follows immediately from the preceeding lemma.

5.2 A multiplication protocol

Using techniques similar to those above, we can also get a protocol for proving that three given commitments c_1, c_2, c_3 contain numbers x_1, x_2, x_3 such that $x_3 = x_1 x_2$. We assume that $c_i = g^{x_i} h^{r_i}$, and as before that the x_i's are numerically smaller than T. Note that then we have $c_3 = c_1^{x_2} h^{r_3 - x_2 r_1}$, i.e., using c_1 as the "base element" for the commitment c_3, it will contain the same value as does c_2 using g as base. So if the prover can convince us of this and also that he can open c_1, it will follow that $x_3 = x_1 x_2$. This is the idea behind the protocol below:

1. P chooses at random $y_1, y \in [0..C(k)T2^k[, s_1, s_2 \in [0..C(k)2^{B+2k}[, s_3 \in [0..C(k)T2^{B+2k}[$ and sends $d_1 = g^{y_1} h^{s_1}, d_2 = g^y h^{s_2}, d_3 = c_1^y h^{s_3}$ to V.
2. V chooses at random e between 0 and $C(k)$ and sends to P.
3. P sends $u_1 = y_1 + ex_1, u = y + ex_2, v_1 = s_1 + er_1, v_2 = s_2 + er_2$ and $v_3 = s_3 + e(r_3 - x_2 r_1)$. V checks that $g^{u_1} h^{v_1} = d_1 c_1^e$, $g^u h^{v_2} = d_2 c_2^e$, and $c_1^u h^{v_3} = d_3 c_3^e$.

Define a relation generator $\mathcal{R}_{\mathsf{mult}}$ as follows: run $\mathcal{G}(1^k)$ to get G, choose $h \in G$ with $C(k)$-rough order, set $g = h^\alpha$ and output $\mathsf{descr}(G), g, h$. Then define the relation $R_{\mathsf{mult}} = \{((c_1, c_2, c_3), (x_1, r_1, b_1, x_2, r_2, b_2, x_3, r_3, b_3)) \mid c_i, b_i \in G, c_i = \mu_i g^{x_i} h^{r_i}, \mu_i^l = 1, \ i = 1, 2, 3\}$. This leads to:

Theorem 3. *If the root assumption is $(t'(k), \epsilon(k))$-secure, the above protocol is a computationally convincing proof of knowledge for $\mathcal{R}$ with knowledge error $\kappa(k)$, time bound $t(k)$ and failure probability $\nu(k)$, where $\nu(k) = 9\epsilon(k)l(k)$, $t(k) < t'(k)/448$ and $\kappa(k) = \max(4/C(k), 448t(k)/t'(k))$. If $-T \leq x_1, x_2, x_3 \leq T$ (as they will be when the prover is honest), the protocol is honest verifier statistical zero-knowledge.*

For the space limitation, we omit the proof, which can be easily derived from the proof of Theorem 2.

6 What is the major difference from the earlier proof in [14]?

For completeness, we briefly indicate here what is mainly different from the earlier work [14] in terms of the proof of soundness. As mentioned above, the main gap we fill here does not appear in the proofs in related works [8, 3, 10, 4]. This is because the gap only appears in the proofs for protocols associated with commitment using plural bases (i.e., $c = g^s h^r$ such as in [14]).

These protocols suggested in [14] are very similar to the ones we suggest here, in particular they have the same 3-move form, with a challenge e from the verifier as the second message. So [14] uses a rewinding argument as we do here, to obtain correct answers from the prover to challenges e, e'. However, a problem occurs in the last part of the proof, which corresponds to the last case in our analysis, that is, Case 2: "$(e-e')$ divides $\alpha(u-u')+(v-v')$". Translated into our notation, we have now $(e-e')$, $u-u'$, and $v-v'$ such that $h^{\alpha(u-u')+(v-v')} = c^{e-e'}$. If $e-e'$ divides both of $(u-u')$ and $(v-v')$, we are essentially done since we then have $c = \mu g^{(u-u')/(e-e')} h^{(v-v')/(e-e')}$ where $\mu^{e-e'} = 1$. In the earlier work [14], it is claimed that the adversary can make Case 2 occur only with negligible probability unless $e-e'$ divides both of $(u-u')$ and $(v-v')$, because he doesn't have enough information about α, where $g = h^\alpha$. However, if $e-e'$ is a small number, then this case may in fact happen with significant probability, even without $e-e'$ dividing both $u-u'$ and $v-v'$. This problem was not taken into account in [14]. Later in the full paper version of [14], it was shown that when the knowledge extractor rewinds P^* and makes him output e, e', it only happens with negligible probability that $e-e'$ is *small*, see [15]. However, even if $e-e'$ is large, there is still a problem: if $e-e'$ has a small prime factor p, there may be a significant probability that p divides $\alpha(u-u')+(v-v')$ whereas it does not divide both $u-u'$ and $v-v'$. The additional idea we provide here essentially fills this gap, and indeed seems necessary for this type of proof to go through.

As for [8, 3, 10, 4], what corresponds to Case 2 is the event "$(e-e')$ divides $(u-u')$"; there is no gap to go to $c = \mu g^{(u-u')/(e-e')}$, where $\mu^{e-e'} = 1$. Hence, the related works above do not suffer from the problem we need to consider.

7 Applying the Scheme in Class Groups and Beyond

We do not give any detailed introduction to class groups here - or more precisely, class groups of quadratic number fields. It is enough to know, that each such group is defined by a single number, the *discriminant* Δ. Given this number, one can choose elements in the group and compute the group and inversion operations. Finding the order of the class group (the class number) from Δ appears to be a hard problem, and is at least as hard as factoring Δ (if Δ is composite). Therefore, root extraction also appears to be a hard problem, and it seems reasonable to conjecture that if Δ is chosen randomly from a large set of values, then the class number will contain large and random prime factors, and will not have a very large factor consisting of only small primes. Various heuristics (but no proofs) supporting this are known. All of this together makes it a reasonable conjecture that class groups constructed from large, random discriminants would satisfy the assumptions we made in the beginning, for some appropriate choice of $C(k)$[4].

There is one difficulty, however: we have assumed that G can be generated such that l_G, the order of the subgroup U of $C(k)$-smooth elements is known. Unfortunately, there is no known way to do this for class groups.

[4] There are some heuristics known that describe how the factorization of a class number can be expected to behave, $C(k)$ should be chosen with this in mind.

One way to solve this is to observe that we have only used l_G in order to verify membership in U. So we can do the following: assume we can choose $C(k)$ and $l(k)$ such that $C(k) > l(k)$ and (as usual) such that $l(k)$ is a polynomial and the order of U is less than $l(k)$. Now we replace l_G in all descriptions and proofs by $l(k)!$. This works because $l(k)!$ is guaranteed to be a multiple of l_G and all its prime factors are at most $l(k)$, and so are less than $C(k)$.

Another possibility is to rely on an additional intractability assumption, namely that given $\mathsf{descr}(G)$, it is hard to find a non-trivial element in U. This seems to be a reasonable assumption in many settings: indeed U is an extremely small subgroup, so a random choice will succeed with negligible probability. Moreover, in the case of class groups with a composite discriminant, finding an element of order 2 is equivalent to factoring the discriminant. With this assumption, all the cases where we needed to know l_G occur with negligible probability, and can be ignored.

References

1. F. Bao: *An efficient verifiable encryption scheme for encryption of discrete logarithm*, In CARDIS'98, LNCS 1820, pp.213–220, 2000.
2. N. Baric and B. Pfitzmann: *Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees*, In EUROCRYPT'97, LNCS 1233, pp.480–494, 1997.
3. F. Boudot and J. Traoré. *Efficient publicly verifiable secret sharing schemes with fast or delayed recovery.* In 2nd ICICS, LNCS 1726, pp.87–102. 1999.
4. F. Boudot: *Efficient Proof that a Committed Number Lies in an Interval*, In Eurocrypt LNCS 1807, Springer, 2000.
5. Boudot: presentation at the rump session of Eurocrypt 2000.
6. M. Bellare and O. Goldreich: *Defining proofs of knowledge*, In Crypto 92.
7. R. Cramer and I. Damgård: *Zero-Knowledge Proofs for Finite Field Arithmetic or: Can Zero-Knowledge be for Free?*, In Crypto 98, LNCS 1462, 1998.
8. A. Chan, Y. Frankel and Y. Tsiounis: *Easy Come - Easy Go Divisible Cash*, In EUROCRYPT'98, pp.561–575 LNCS 1403, 1998.
9. J. Camenisch and M. Michels: *Proving in Zero-Knowledge that a Number Is the Product of Two Safe Primes*, In Eurocrypt'99 pp.107–122 LNCS 1592, 1999.
10. J. Camenisch and M. Michels: *Separability and Efficiency for Generic Group Signature Schemes*, In CRYPTO'99 pp.413–430, LNCS 1666, 1999.
11. J. Camenisch and M. Michels: *Proving in Zero-Knowledge that a Number Is the Product of Two Safe Primes*, Tech. Report RS-98-29, BRICS, 1999.
12. I. Damård: *Practical and Provably Secure release of a Secret and Exchange of Signatures*, J.Cryptology, vol. 8, pp.201-222, 1995.
13. E. Fujisaki: *A simple Approach to Secretly Sharing a Factoring Witness in a Publically-Verifiable Manner*, IEICE Trans. Fund., E85-A, vol.5, May 2002.
14. E. Fujisaki and T. Okamoto: *Statistical Zero-Knowledge Protocols to prove Modular Polynomial Relations*, In Crypto 97, LNCS 1294, 1997.
15. E. Fujisaki and T. Okamoto: *Statistical Zero-Knowledge Protocols to Prove Modular Polynomial Relations*, in IEICE Trans. Fund., E82-A, vol.1 pp. 81–92, Jan. 1999.
16. Goldwasser, Micali and Rackoff: *The knowledge complexity of interactive proof systems*, SIAM J.Computing, vol. 18, pp.186-208, 1989.

17. T. Pedersen: *Non-Interactive and Information Theoretic Secure Verifiable Secret Sharing*, In Crypto 91, LNCS 576, pp. 129–140.
18. P. Paillier:*Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, In Eurocrypt'99, LNCS 1592, pp. 223–238, 1999.
19. T. Okamoto and S. Uchiyama: *A New Public-Key Cryptosystem as Secure as Factoring* In Eurocrypt 98, LNCS 1403, 1998.

A The **Rewind** procedure

We are given a prover P^* who sends a message d, receives a challenge chosen randomly among C possibilities, and returns an answer z that may or may not be correct. We are given that the probability of a correct answer taken over P^* coins and the choice of e is at least $\epsilon > 4/C$. We want to find correct answers to two different e-values for a given d as efficiently as possible.

Of course, the idea is to run the prover, and use rewinding to try to make him answer two different challenges correctly. But to run him, we need to supply random coins. Although we know that the average success probability is ϵ, we do not know that P^* is equally successful with any random input. To get a better view of this, let H be a matrix with a row for each possible set of random coins for P^*, and one column for each possible challenge value. Write 1 in an entry if P^* answers correctly with the corresponding random choices and challenge, and 0 otherwise. Using P^* as black-box, we can probe any entry we want in H, and our goal can be rephrased to: find two 1's in the same row. What we know is that the ϵ equals the fraction of 1-entries in H.

It is now apparent that we cannot just search for a 1-entry and then keep looking for another 1 in the same row: if we stumbled across the only 1 in that row, we will never finish. Consider instead the following algorithm Alg:

1. Probe random entries in H until a 1 is found.
2. Then start the following two processes in parallel, and stop when either one stops:
   - Pr_1 Probe random entries in the row in which we found a 1 before, until another 1-entry is found.
   - Pr_2 Repeatedly flip a coin that comes out heads with probability ϵ/w, for some constant integer w (we show how to choose w later), until you get heads. This can be done by probing a random entry in H and choosing a random number among $1, 2, ..., w$ - you output heads if the entry was a 1 and the number was 1.

This algorithm runs in expected time at most w/ϵ, recall that we count access to P^* as one step. Define a row to be *heavy* if it contains a fraction of at least $\epsilon/2$ 1's. By a simple counting argument, you can see that at least half of the 1's are located in heavy rows. Given that Pr_1 runs in a heavy row, the probability that a probe will succeed is at least $\frac{C\epsilon/2-1}{C}$ so the expected number of probes it makes is $T(\epsilon) = C/(C\epsilon/2 - 1)$. If $\epsilon \geq 4/C$, then $T(\epsilon) \leq 2/\epsilon$. Moreover, the probability that Pr_1 runs for more time than $2T(\epsilon)$ is at most $1/2$. Assume we choose w large enough, so that $Pr2$ finishes later than $2T(\epsilon)$ with probability of at least $1/2$. It is straightforward to see that $w = 7$ is sufficient. Then, given that the row we use is heavy, we have probability of at least $1/4$ of success, and hence overall probability $1/8$.

Therefore, our Rewind procedure simply repeats Alg until there is success, the expected number of times it will have to do so is 8, and hence the expected total time is $56/\epsilon$.

B Further Extensions

There are many possible variants or extensions with some differences: For example, when $T_0 < |x|$ is public, we can reduce the computational amount for the protocol by using $T - T_0$ instead of T. In the rest of this section, we briefly mention to extensions, though not many due to the space limitation, to non-Abelian groups and verifiable encryptions, where we think the latter might not be so trivial.

For non-Abelian group G, we can get a similar result if the assumptions are modified as follows: (1) there is a element h in G such that its order is C-rough whereas $l_G \triangleq [G :< h >]$ is C-smooth, and (2) given $\mathsf{descr}(G)$ and random $Y \in G$, it is difficult to find $e > 1, X \in G$ such that $Y = \mu_1 X^e \mu_2$ and $\mu_1^{l_G} = \mu_2^{l_G} = 1$. Define $\mathsf{commit}_{pk}(s, r) = g^s h^r$ for $g \in< h >$, but allow the committer to send (s', r') when opening commitment c so long as $(g^{s'} h^{r'}/c)^{l_G} = 1$. The proofs above similarly goes through considering l_G-th power of any element in G belongs to $< h >$.

To make verifiable encryption, we use the Okamoto-Uchiyama encryption [19]. Suppose that p, q are primes such that $(p-1)/2, (q-1)/2$ are C-rough. Define $\mathsf{QR}(X) \triangleq \{x \in \mathbb{Z} \mid \exists y \in \mathbb{Z} \text{ s.t. } y^2 = x \pmod X\}$. Let $H = \mathsf{QR}(n) \subset G = \mathsf{QR}(q) \subset (\mathbb{Z}/n\mathbb{Z})^\times$ where $n = p^2 q$. Since $(\frac{x}{n}) = (\frac{x}{p})^2 (\frac{x}{q})$, one can efficiently check that x belongs to G by computing the Jacobian symbols over n. Then g, h are chosen as follows: Pick up at random $h_0 \in H$ such that $p|\mathsf{ord}(h)$. Set $g = h_0^\alpha$ and $h = h_0^n$. We have $l_G = 4$. The commitment $c = g^x h^r$ $(0 < x < p)$ is not statistical hiding but one can still think of it as computational hiding (so long as the OU encryption is semantically secure). The associated protocols above can be applied to this new commitment without any modification, which makes this verifiable. Actually in this case, the verifier can be convinced that the committer can only open commitments that belong to H (because if the committer can show any non-trivial μ such that $\mu^4 = 1$, he can factor n).

An application of this verifiable encryption appears in [13]. A "light" version of this verifiable encryption, using $c = g^s \bmod n$, appears in [1].