# Minimal Assumptions for Efficient Mercurial Commitments[*]

Yevgeniy Dodis[†]

**Abstract**

Mercurial commitments were introduced by Chase et al. [8] and form a key building block for constructing zero-knowledge sets (introduced by Micali, Rabin and Kilian [27]). Unlike regular commitments, which are strictly binding, mercurial commitments allow for certain amount of (limited) freedom. The notion of [8] also required that mercurial commitments should be equivocable given a certain trapdoor, although the notion is interesting even without this condition. While trivially implying regular (trapdoor) commitments, it was not clear from the prior work if the converse was true: Chase et al. [8] gave several constructions of mercurial commitments from various incompatible assumptions, leaving open if they can be built from any (trapdoor) commitment scheme, and, in particular, from any one-way function. We give an affirmative answer to this question, by giving two simple constructions of mercurial commitments from any trapdoor bit commitment scheme. By plugging in various (trapdoor) bit commitment schemes, we get *all* the efficient constructions from [27, 8], as well as several immediate new constructions.

Our results imply that (a) *mercurial commitments can be viewed as surprisingly simple variations of regular (trapdoor) commitments* (and, thus, can be built from one-way functions and, more efficiently, from a variety of other assumptions); and (b) *the existence of zero-knowledge sets is equivalent to the existence of collision-resistant hash functions* (moreover, the former can be efficiently built from the latter and trapdoor commitments). Of independent interest, we also give a stronger and yet much simpler definition of mercurial commitments than that of [8] (which is also met by our constructions). Overall, we believe that our results eludicate the notion of mercurial commitments, and better explain the rational following the previous constructions of [27, 8].

[*]This is an independent part of the joint paper with Dario Catalano and Ivan Visconti from TCC'06 [6].

[†]Department of Computer Science, New York University, 251 Mercer Street, New York, NY 10012, USA. Email: `dodis@cs.nyu.edu`

## 1 Introduction

Commitment schemes are important cryptographic primitives. They allow one party to commit to some value $v$ so that $v$ is kept secret from the rest of the world (this is called *hiding*), and yet everybody knows that the value $v$ is uniquely defined at the time $v$ was committed (this is called *binding*). In particular, binding ensures that the party cannot announce the commitment first, and then decide later how to open it depending on the circumstances. In this sense, commitment schemes force the party to fully decide on what he is committing to.

In Eurocrypt 2005, Chase et al. [8] introduced an intriguing variant of commitments called *mercurial commitments*. The main difference comes from the fact that mercurial commitments allow for a small, and yet noticeable relaxation of the strict binding property of regular commitments. Namely, they allow for a two-stage opining protocol. In the *soft-open stage* the committer can claim that "if I committed to anything at all, then this value is $m$", while in the hard-opening stage he would indeed declare that "Yes, I really committed to the value $m$." In particular, any committed value $c$ can either be both soft- and hard-opened only to one (correct!) message $m$, or can be soft-opened to arbitrary messages, but then it cannot be hard-opened at all! Moreover, the committer must decide before forming the commitment which one of the two cases suits him better: to commit to only one value, or not to commit to anything at all. Although this is seemingly not much better than regular commitments, the extra freedom of the committing party comes from the fact that by showing a soft-opening of his commitment to some value $m$, the receivers still cannot tell if $m$ was really committed to by $c$, or if $c$ was simply a "non-commitment" to anything (and the committer might be just going around and soft-opening $c$ to arbitrary values $m'$). The receivers are sure, however, that it is impossible to hard-open $c$ to any $m' \neq m$.

Chase et al. [8] distilled the above natural primitive to abstract away a relatively complicated (but efficient!) construction of *zero-knowledge sets* by Micali et al. [27]. Such ZK sets allow one to commit to some secret set $S$ over some universe, and then to be able to non-interactively prove statements of the form $x \in S$ and $x \notin S$, and yet no other information (which cannot be deduced from the inclusions/exclusions above) about $S$ is leaked — not even its size! With the abstraction of mercurial commitments, Chase et al. [8] obtained an elegant and easy-to-follow general "explanation" of the construction from [27]. Namely, they showed that the construction of [27] is an instance of a general construction of ZK sets from *any* mercurial commitment scheme and any collision-resistant hash function.

PLAIN VS. TRAPDOOR MERCURIAL COMMITMENTS. We remark that to match a very strong zero-knowledge definition of ZK sets from [27], Chase et al. [8] had to require that mercurial commitments satisfy the following "equivocation" property: there exists some trapdoor information $msk$ (ordinarily not available to anybody) which enables one to completely destroy all the binding properties of mercurial commitments. Namely, using $msk$ one can construct fake commitments, which look just like regular commitments and yet can be soft- or hard-opened to completely arbitrary values. (This is very similar to the notion of regular *trapdoor commitments* [4], where the knowledge of the corresponding trapdoor key can enable somebody to create fake regular commitments which can be opened to any message.) As already observed by [8], this strong equivocation property does not seem to be inherent for the "plain" primitive of mercurial commitments, but they chose to insist on this extra property since it was need for their main application. Since we believe that mercurial commitments are also interesting without equivocation, in our results we will distinguish between *plain* and *trapdoor* mercurial commitments. (Although our results described below will hold equally naturally for either case.) Indeed, we observe that one can define a weaker notion of ZK sets, which we informally call *indistinguishable sets*, which have the same functionality as ZK sets, but the privacy property is relaxed to only state that for any two sets and any sequence of inclusion/exclusion assertions which does not "separate" these sets, seeing the proofs of the corresponding assertions does not allow one to distinguish between these two sets. (This is somewhat similar to the distinction between witness indistinguishable [16] and ZK proofs [20].) And then it is easy to see that the same generic construction from [8] would give indistinguishable sets when applied to plain mercurial commitments. To summarize, we believe that both plain and trapdoor mercurial commits are useful and deserve investigation.

MINIMAL ASSUMPTIONS FOR MERCURIAL COMMITMENTS. Having introduced a new cryptographic primitive, it is always very important to understand where it lies in the hierarchy of cryptographic assumptions. Towards this goal, [8] gave a general construction of (trapdoor) mercurial commitments from non-interactive zero-knowledge proofs (NIZK) for NP (which are known, for example, to be implied by trapdoor permutations). However, this construction is mainly of theoretical interest, since it is very inefficient in practice. They also gave a more efficient[1] construction of mercurial commitments from an even stronger assumption of claw-free permutations [21]. On the other hand, [8] observed that (trapdoor) mercurial commitments are similar and trivially imply (trapdoor) regular commitments,[2] although they pointed out some important differences as well. Thus, the following two questions were left open:

**Question 1:** *What minimal cryptographic assumptions are sufficient for plain/trapdoor mercurial commitments?*

**Question 2:** *Can plain/trapdoor mercurial commitments be (efficiently) built from plain/trapdoor commitments?*

Our first result resolves these questions in a surprisingly simple fashion. We show a very simple and efficient construction of (bit) plain/trapdoor mercurial commitments from any bit plain/trapdoor regular commitment. The construction is a very simple generalization of the claw-free construction from [8], but since trapdoor commitments are (in principle) equivalent to one-way functions, we get

**Theorem 1** *There exists a simple and efficient construction of bit plain/trapdoor mercurial commitments from bit plain/trapdoor regular commitments. In particular, mercurial commitments of either kind exist if and only if one-way functions exist.*

EFFICIENCY? Having resolved the question of feasibility, we can turn to the question of efficiency. Of course, we can plug in various efficient bit trapdoor commitment schemes to our previous construction, but this will only result in bit-by-bit constructions for long messages, which is pretty inefficient for practical use (e.g., for the ZK sets application). On the other hand, Chase et al. [8] gave two efficient constructions for long messages based on specific number-theoretic constructions (discrete log and factoring; the discrete log construction was implicit in [27]). Examining these constructions, one can see that that there seems to be some kind of similarity between them, although it is not obvious exactly where this similarity comes from. Also, it is relatively hard to understand why each construction is really secure, without going into the details of the proof. Motivated by this, we ask

**Question 3:** *Is there an efficient and yet reasonably* **general** *construction of plain/trapdoor mercurial commitments, which would abstract and explain the efficient number-theoretic constructions from [8]?*

Our second and main result gives a surprisingly general answer to this question. Namely, we present a construction which directly transforms a plain/trapdoor bit commitment $\mathcal{C}$ into an efficient and (typically) multi-bit plain/mercurial commitment $\mathcal{C}'$. Namely, we still base it on general plain/trapdoor commitment, just like in Theorem 1. However, a small catch is that we will need to assume an extra property from $\mathcal{C}$ (see Section 2.1 for a definition of $\Sigma$-protocol):

**Theorem 2** *Assume $\mathcal{C}$ is a plain/trapdoor bit commitment which has an efficient $\Sigma$-protocol $\Pi$ proving that one knows a witness $d$ that a given (regular) commitment $c$ can be opened to $0$.[3] Then one one can construct an efficient plain/trapdoor mercurial commitment $\mathcal{C}'$ whose message space is equal to the challenge space of $\Pi$.*

Thus, to get message-efficient constructions, it will be important to design "challenge-efficient" $\Sigma$-protocols for our plain/trapdoor commitment schemes. While such $\Sigma$-protocol's $\Pi$ in principle (see Theorem 4 below) can always be built from one-way functions, in general this will not outperform the simple construction in Theorem 1. However,

[1]The construction is efficient when committing to a bit. To commit to longer messages, one has to use it in a bit-by-bit fashion.

[2]Both plain and trapdoor commitments are known to be equivalent to the minimal assumption of one-way functions; see Section 2.2.

[3]As explained in Section 4 proving this theorem, we will need a slight extra property (*) from such $\Sigma$-protocols, but it will always hold in any practical construction we are aware of. So we omit it from this statement.

the utility of this transformation comes from the fact that all number-theoretic (trapdoor) bit commitment schemes have very efficient $\Sigma$-protocols, and usually with rich challenge spaces. Plugging in various such commitment schemes with efficient protocols, we get many efficient constructions of mercurial commitments. In particular, both the discrete log and the factoring construction of [8] become special cases of our general transformation, when applied to an appropriate commitment scheme! And several new constructions can be obtained as well (e.g., from RSA and Paillier [30] assumptions, as well as new discrete log and factoring constructions; see Section 4.1). More generally, we also believe that our construction is much easier to understand and sheds more light on why the previous number-theoretic constructions where built in this way.

SIMPLER DEFINITION. As another small contribution, by *strengthening* the definition of trapdoor mercurial commitments as compared to the definition of [8], we considerably simplified the equivocation property of mercurial commitments. Since our constructions all satisfy the stronger definition, and it results in easier and shorter proofs, we believe our strengthening is justified and could be of independent interest.

IMPLICATION TO ZK SETS. Recall that indistinguishable/ZK sets can be efficiently constructed from any plain/trapdoor mercurial commitment scheme and a collision-resistant hash function (CRHF). Chase et al. [8] also made a simple observation that ZK sets imply the existence of CRHFs. Using Theorem 1, Theorem 2, and the fact that CRHFs imply both one-way functions (and, thus, plain/trapdoor commitments) and efficient *plain* commitment schemes [12, 24], we immediately obtain:

**Theorem 3** *The existence of ZK (and, thus, indistinguishable) sets is equivalent to the existence of CRHFs. In fact, ZK sets can be* efficiently *constructed from CRHFs and trapdoor bit commitment schemes, while indistinguishable sets can be efficiently constructed using CRHFs alone (by also building commitments out of them). Moreover, the constructions become even more efficient if the commitment scheme in question has a challenge-efficient $\Sigma$-protocol needed for Theorem 2.*

## 2 Definitions

### 2.1 $\Sigma$-Protocols

Let $\mathcal{R} = \{(x, w)\}$ be some $\mathsf{NP}$-relation (i.e., it is efficiently testable to see if $(x, w) \in \mathcal{R}$ and $|w| \leq \mathsf{poly}(|x|)$). We usually call $x$ the input, and $w$ — the witness (for $x$). Consider a three move protocol run between a PPT prover $P$, with input $(x, w) \in R$, and a PPT verifier $V$ with input $x$, of the following form. $P$ chooses a random string $r_p$, computes $a = \mathsf{Start}(x, w; r_p)$, and sends $a$ to $V$. $V$ then chooses a random string $e$ (called "challenge") from some appropriate domain $E$ (see below) and sends it to $P$. Finally, $P$ responds with $z = \mathsf{Finish}(x, w, e; r_p)$. The verifier $V$ then computes and returns a bit $b = \mathsf{Check}(x, a, e, z)$. We require that $\mathsf{Start}$, $\mathsf{Finish}$, and $\mathsf{Check}$ be polynomial-time algorithms, and that $|e| \leq \mathsf{poly}(|x|)$. Such a protocol (given by procedures $\mathsf{Start}, \mathsf{Finish}, \mathsf{Check}$) is called a $\Sigma$-*Protocol* for $\mathcal{R}$ if it satisfies the following properties, called completeness, special soundness, and special honest-verifier zero-knowledge:

- **Completeness:** If $(x, w) \in R$ then the verifier outputs $b = 1$ (with all but negligible probability).
- **Special Soundness:** There exists a PPT algorithm $\mathsf{Extract}$, called the (knowledge) extractor, such that it is computationally infeasible to produce an input tuple $(x, a, e, z, e', z')$ such that $e \neq e'$ both lie in the proper "challenge" domain, $\mathsf{Check}(x, a, e, z) = \mathsf{Check}(x, a, e', z') = 1$, and yet $\mathsf{Extract}(x, a, e, z, e', z')$ fails to output a witness $w$ such that $(x, w) \in R$. Intuitively, if some prover can correctly respond to two different challenges $e$ and $e'$ on the same first flow $a$, then the prover must "know" a correct witness $w$ for $x$ (in particular, $x$ has a witness).
- **Special HVZK:** There exists a PPT algorithm $\mathsf{Simul}$, called the simulator, such that for any $(x, w) \in R$ and for any fixed challenge $e$, the following two distributions are computationally indistinguishable. The first distribution $(x, a, e, z)$ is obtained by running an honest prover $P$ (with some fresh randomness $r_p$) against a ver-

ifier whose challenge is fixed to $e$. The second distribution $(x, a, e, z)$ is obtained by computing the output $(a, z) \leftarrow \mathsf{Simul}(x, e)$ (with fresh randomness $r_s$). Intuitively, this says that for any a-priori fixed challenge $e$, it is possible to produce a protocol transcript computationally indistinguishable from an actual run with the prover (who knows $w$).

It is easy to see that the standard zero-knowledge protocol for the Hamiltonian Cycle [15, 22] language is a (binary challenge) $\Sigma$-protocol. Therefore,

**Theorem 4 ([22, 15])** *Any* $\mathsf{NP}$*-relation* $\mathcal{R}$ *has a (binary challenge)* $\Sigma$*-protocol if secure commitment schemes exist (in particular, if one-way functions exist).*

Of course, we will see and crucially exploit the fact that many natural specific languages have much more efficient $\Sigma$-protocols (see below). We also notice that, aside from computational efficiency, a good quality measure for a given $\Sigma$-protocol is the size of its challenge space $E$ (the larger the better). One reason for this dependency comes because the special soundness property easily implies that if a malicious prover does not "know" a valid witness $w$ for $x$, then he can succeeds in fooling the verifier with probability at most (only negligibly better than) $1/|E|$. In our application, we will also see that the large size of $E$ will also naturally translate to more efficient solutions, and we will therefore strive to use "challenge-efficient" $\Sigma$-protocols.

GENERALIZATIONS. First, we will allow $\mathcal{R}$ to depend on some honestly generated public parameter $pk$ (known to everybody after generation); e.g. the standard discrete-log relation would be $\mathcal{R}_{p,g}(x, w) = 1$ if and only if $x = g^w \bmod p$, where the prime $p$ and the generator $g$ could be randomly generated. In this case the corresponding properties of the $\Sigma$-protocol should computationally hold over the choice of such parameters. However, for one of our applications we will require an even stronger technical property. Namely, we will say that a family of relations $\{\mathcal{R}_{pk}\}$ has a $\Sigma$-protocol which is *strongly hiding w.r.t. instance generation procedure* $\mathcal{P}$ if the special HVZK property holds even in the following experiment. $\mathcal{P}$ produces $(pk, x, w, \mathcal{I})$, where $pk$ is the public key for $\mathcal{R}$, $x$ is the input, $w$ is the witness, and $\mathcal{I}$ is some side information available to attacker. Then we either give to the distinguisher a tuple $(\mathcal{I}, pk, x, a, e, z)$ obtained by having the prover run the real protocol with $x$ and $w$, or where $(a, z)$ is produced by the simulator $\mathsf{Simul}_{pk}(x, e)$. To put it differently, the side information $\mathcal{I}$ does not help the distinguisher to break the special HVZK property. We notice that, essentially all of the practical $\Sigma$-protocols known (including all the ones we will actually consider) will satisfy the statistical HVZK property, in which case they will be strongly-hiding w.r.t. any $\mathcal{P}$. Also, the generic protocol from Theorem 4 will also be strongly-hiding w.r.t. any efficient procedure $\mathcal{P}$ which only depends on the public parameters of the commitments used inside the protocol. This, once again, includes essentially all interesting procedures (including the specific one we will need later). In other words, for all practical purposes this extra property is just a technicality we need for the proof to go through.

As a second, orthogonal generalization, we can also consider "auxiliary-input" $\Sigma$-protocols, where in order to run the protocol, the prover $P$ might need some extra information $\mathsf{aux}$ satisfying some property (which, presumably can be generated together with $(x, w)$), in addition to $w$. Notice, $w$ alone is enough to allow for verification that $(x, w) \in \mathcal{R}$, so $\mathsf{aux}$ is only needed by the prover to fulfill his completeness requirement (in particular, the simulator does not need to know $\mathsf{aux}$ and special soundness and HVZK stay the same as before).

EFFICIENT $\Sigma$-PROTOCOLS. We briefly survey the following efficient $\Sigma$-protocols which we will use in the sequel. (The exact details will not be crucial for our purposes, so we will not present them here.) We notice that most of them will be unconditional: the security assumption behind the relation (such as discrete log) will be used later in the application; for example, in claiming that the hypothetical extraction of the witness contradicts the corresponding assumption.

The Schnorr $\Sigma$-protocol [32] allows one to unconditionally prove the knowledge of the discrete log in cyclic groups of prime order. A less known fact [18, 9] is that (a slightly modified)[4] Schnorr protocol also works over the subgroup of quadratic residues $Q_n$ over $\mathbb{Z}_n^*$, where $n$ is the product of two safe primes. Interestingly, unlike in prime order groups, where the special soundness holds unconditionally, here it will hold computationally under the strong RSA assumption. In both of these cases the challenge space is exponential.

Very similar to Schnorr protocol, Gilliou-Quisquater (GQ) [23] protocol proves the knowledge of the $e$-th root over $\mathbb{Z}_n^*$ (i.e., solution to RSA), where $\gcd(e, \varphi(n)) = 1$ and $n$ is the produce of two safe primes. Here, however, the challenge space should be smaller than the exponent $e$, so this protocol is challenge-efficient only if $e$ is large (which is typically required when this protocol is used).

The Fiat-Shamir protocol is an unconditional binary-challenge $\Sigma$-protocol proving the knowledge of the square root over $\mathbb{Z}_n^*$, where $n$ is the product of two primes. One way to make it challenge-efficient is to repeat it in parallel, but this is computationally inefficient. A better way is to use the elegant technique of Ong-Schnorr [29], at the expense of working over the set of quadratic residues $Q_n$, requiring $n$ to be a Blum integer, and, more crucially, requiring an auxiliary witness to the prover. Namely, in order to make the challenge space to be of size $2^\ell$, the prover not only needs to know a square root of the input $x \in Q_n$, but also the $2^\ell$-root root $u \in Q_n$ of $x$ (which is well defined when $n$ is a Blum integer): see Lemma 3.1 in [1] explicitly stating the special soundness of this protocol. Of course, to run this protocol in practice one would first pick $u$ and then set $w = u^{2^{\ell-1}} \bmod n$ (by repeated squaring) and $x = w^2 \bmod n$.

All the above mentioned protocols have statistical special HVZK, so they always satisfy strong-hiding. To summarize, natural relations arising from well established cryptographic assumptions have very computationally and challenge-efficient $\Sigma$-protocols.

## 2.2 Commitments and Trapdoor Commitments

COMMITMENTS. A (non-interactive) commitment scheme consists of four efficient algorithms: $\mathcal{C} = (\mathsf{Com\text{-}Gen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Ver})$. The generation algorithm $\mathsf{Com\text{-}Gen}(1^k)$, where $k$ is the security parameter, outputs a public commitment key $pk$ (possibly empty, but usually consisting of public parameters for the commitment scheme). Given a message $m$ from the associated message space $\mathcal{M}$ (e.g., $\{0,1\}^k$, although we will mainly concentrate on bit commitments), $\mathsf{Com}_{pk}(m; r)$ (computed using the public key $pk$ and additional randomness $r$) produces a commitment string $c$ for the message $m$. We will sometimes omit $r$ and write $c \leftarrow \mathsf{Com}_{pk}(m)$. Similarly, the opening algorithm $\mathsf{Open}_{pk}(m; r)$ (which is supposed to be run using the same value $r$ as the commitment algorithm) produces a decommitment value $d$ for $c$. Finally, the verification algorithm $\mathsf{Ver}_{pk}(m, c, d)$ accepts (i.e., outputs 1) if it thinks the pair $(c, d)$ is a valid commitment/decommitment pair for $m$. We require that for all $m \in \mathcal{M}$, $\mathsf{Ver}_{pk}(m, \mathsf{Com}_{pk}(m; r), \mathsf{Open}_{pk}(m; r)) = 1$ holds with all but negligible probability.

We remark that without loss of generality we could have assumed that the opening algorithm simply outputs its randomness $r$ as the decommitment, and the verification algorithm simply checks if $c = \mathsf{Com}_{pk}(m; r)$. However, we will find our more general notation more convenient for our purposes. When clear form the context, we will sometimes omit $pk$ from our notation. Regular commitment schemes have two security properties:

- **Hiding:** No PPT adversary (who knows $pk$) can distinguish the commitments to any two message of its choice: $\mathsf{Com}_{pk}(m_1) \approx \mathsf{Com}_{pk}(m_2)$. That is, $\mathsf{Com}_{pk}(m)$ reveals "no information" about $m$.
- **Binding:** Having the knowledge of $pk$, it is computationally hard for the PPT adversary $\mathcal{A}$ to come up with $c, m, d, m', d'$ such that $(c, d)$ and $(c, d')$ are valid commitment pairs for $m$ and $m'$, but $m \neq m'$ (such a tuple is said to cause a *collision*). That is, $\mathcal{A}$ cannot find a value $c$ which it can open in two different ways.

[4] In particular, the prover works over the integers instead of over $\mathbb{Z}_{|Q_n|}$, since he does not know $|Q_n|$. Because of that the special HVZK guarantee is statistical here rather than perfect.

Commitments are known to be theoretically equivalent to one-way functions [28, 25]. However, efficient commitments can be built from collision-resistant hash functions [12, 24], and many number-theoretic assumptions (such as factoring, discrete log and RSA, and Paillier [30]; see below). In fact, most of these number-theoretic construct give a stronger kind of commitment — called trapdoor commitment — which we explain next.

TRAPDOOR COMMITMENTS. A (non-interactive) trapdoor commitment scheme consists of six efficient algorithms: $\mathcal{C} = (\mathsf{TrCom\text{-}Gen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Ver}, \mathsf{Fake}, \mathsf{Equiv})$. The generation algorithm $\mathsf{TrCom\text{-}Gen}(1^k)$, where $k$ is the security parameter, outputs a public commitment key $pk$ and and a secret *trapdoor key* $sk$. Once $pk$ is fixed, the meaning of Com, Open and Ver is exactly the same as for regular commitments. In particular, we will require that these algorithms satisfy the usual hiding and binding properties of the commitment schemes.

The trapdoor key $sk$ is used in the algorithms Fake and Equiv to break the binding property of commitments. Namely, $\mathsf{Fake}_{sk}(;r)$ (which takes no input except for randomness $r$) produces "fake" commitment $c$, initially not associated to any message $m$. On other other hand, for any message $m$, $\mathsf{Equiv}_{sk}(m;r)$ (which is supposed to be run using the same value $r$ as the fake commitment algorithm) produces a "fake decommitment" value $d$ for $c = \mathsf{Fake}_{sk}(;r)$. In particular, we require that such fake $(c, d)$ still satisfy the verification equation: for all $m \in \mathcal{M}$, $\mathsf{Ver}_{pk}(m, \mathsf{Fake}_{sk}(;r), \mathsf{Equiv}_{sk}(m;r)) = 1$ holds with all but negligible probability. Even stronger, we require that

- **Equivocation:** for any $m \in \mathcal{M}$ (chosen by the adversary), a "true" commitment tuple $(m, \mathsf{Com}_{pk}(m;r)$, $\mathsf{Open}_{pk}(m;r))$ should look computationally indistinguishable (over $r$) from the fake tuple $(m, \mathsf{Fake}_{sk}(;r)$, $\mathsf{Equiv}_{sk}(m;r))$. More importantly, we require that these distributions should look indistinguishable even if the distinguisher knows not only the commitment key $pk$, but also *the trapdoor key* $sk$ (we will explain the rational for this shortly)!

We notice that equivocation easily implies that trapdoor commitments satisfy the usual hiding property of commitments (since all commitments $\mathsf{Com}_{pk}(m)$ are indistinguishable from a single distribution $\mathsf{Fake}_{sk}()$): in fact, this indistinguishability holds even if the distinguisher knows $sk$! Thus, binding and equivocation are enough to argue the security of trapdoor commitment schemes.

We briefly give the rational of why we need such a strong equivocation property. This is done for the purposes of *composition*. Indeed, we would like to argue that given several "real" pairs $(c, d)$, we can replace all of them by the corresponding "fake" pairs $(c', d')$, without anybody "noticing". However, the standard left-to-right hybrid argument requires us to be able to generate not only the "real left-pairs" $(c, d)$, which we can do using $pk$, but also "fake right-pairs" $(c', d')$, and this we cannot do without the knowledge of $sk$. Requiring the indistinguishability to hold even with the knowledge of $sk$ resolves this problem, and gives us all the natural composition properties.

CONSTRUCTIONS. There are many constructions of trapdoor commitments (and each of them also gives a regular commitment, of course). For example, efficient trapdoor commitments exist based on a variety of number-theoretic assumptions: factoring [26, 31], discrete log [3, 4]), RSA (combining [15, 23]), Paillier [5, 11]. In fact, some of these schemes (e.g., those based on discrete log and RSA) are special cases of a beautiful general construction by Feige and Shamir [15]. This construction *efficiently* transforms any $\Sigma$-protocol corresponding to a "hard" language in $\mathsf{NP}$ into a trapdoor commitment scheme. In particular, since we mentioned that all of $\mathsf{NP}$ has such $\Sigma$-protocols if one-way functions exists (see Theorem 4), and the latter also imply that some languages in $\mathsf{NP}$ are "hard", one can in principle construct a trapdoor commitment scheme *from any one-way function* (see sec. 4.9.2.3 of [19]). We note that the message space for the resulting trapdoor commitment will be exactly the challenge space of the corresponding $\Sigma$-protocol, which, once again, demonstrates why we want to construct challenge-efficient $\Sigma$-protocols.[5] Quite interestingly, this construction *of* trapdoor commitments will be somewhat reminiscent to our main construction *from* trapdoor commitments (possessing a certain $\Sigma$-protocols; see Section 4), although this seems to be more of a

[5]Of course, since both of our generic mercurial commitment constructions only use *bit* commitments, even binary $\Sigma$-protocol's for hard languages easily suffice for our main purpose.

coincidence.[6]

We also mention another, less general construction [26] of trapdoor commitments from claw-free permutation pairs [21]. This construction is only efficient for bit trapdoor commitments (which, once again, are sufficient for us). Looking at various known claw-free permutation constructions (e.g., see [13] for such a list), we immediately get efficient bit trapdoor commitment constructions from various assumptions, such as the already mentioned constructions from factoring [26], Paillier [11] and the bit-version of the discrete log construction of [3, 4]. In regards to discrete log, we finally mention the following "ad-hoc" construction of trapdoor bit commitments which we will later use. The public key consists of two random generators $g$ and $h = g^x$ of some prime order $q$ cyclic group $\mathbb{G}$, where the discrete log is hard (here $x$ is a random non-zero element of $\mathbb{Z}_q$), while the trapdoor key is $x$. To commit to $0$, one computes $g^{r_0}$ (for random non-zero $r_0 \in \mathbb{Z}_q$), while to commit to $1$ one similarly computes $h^{r_1}$. The openings are $r_0$ and $r_1$, respectively. To break binding one needs to satisfy $g^{r_0} = h^{r_1}$, which means that one can compute $x = r_0 r_1^{-1} \bmod q$ (and this contradicts discrete log). On the other hand, if $x$ is known, it is trivial to open a "fake" commitment $h^{r_1}$ both to $1$ (by simply presenting $r_1$) and to $0$ (by presenting $r_1 x \bmod q$).

## 2.3 Mercurial Commitments

We now define mercurial commitments introduced by Chase et al. [8]. Our definition will be similar, but *stronger* than the definition from [8]. There are two reasons for making the change. First, all the efficient construction in [8] and here will anyway satisfy the stronger definition. More importantly, by making our definition stronger we will also make it noticeably simpler (and shorter!) than the definition of [8]. More detailed comparison will be given later in the section.

PLAIN MERCURIAL COMMITMENTS. Such commitment schemes consist of seven efficient algorithms: $\mathcal{C} = (\mathsf{MCom\text{-}Gen}, \mathbb{H}\mathsf{Com}, \mathbb{H}\mathsf{Open}, \mathbb{H}\mathsf{Ver}, \mathbb{S}\mathsf{Com}, \mathbb{S}\mathsf{Open}, \mathbb{S}\mathsf{Ver})$. The first four algorithms $(\mathsf{MCom\text{-}Gen}, \mathbb{H}\mathsf{Com}, \mathbb{H}\mathsf{Open}, \mathbb{H}\mathsf{Ver})$ follow the syntax (and the functionality!) of regular commitment schemes (see Section 2.2). Namely, generation algorithm $\mathsf{MCom\text{-}Gen}(1^k)$, where $k$ is the security parameter, outputs a public mercurial commitment key $mpk$. Given a message $M \in \mathcal{M}$, the *hard-commit* algorithm $\mathbb{H}\mathsf{Com}_{mpk}(M; R)$ produces a *hard-commitment* string $C$ for $M$. We will sometimes write $C \leftarrow \mathbb{H}\mathsf{Com}_{mpk}(M)$. Similarly, the *hard-opening* algorithm $\mathbb{H}\mathsf{Open}_{mpk}(M; R)$ (which is supposed to be run using the same value $R$ as the hard-commit algorithm) produces a *hard-decommitment* value $\pi$ for $C$. Finally, the *hard-verification* algorithm $\mathbb{H}\mathsf{Ver}_{mpk}(M, C, \pi)$ accepts (i.e., outputs 1) if it thinks $\pi$ proves that $C$ is indeed a valid hard-commitment to $M$. We require that for all $M \in \mathcal{M}$, $\mathbb{H}\mathsf{Ver}_{mpk}(m, \mathbb{H}\mathsf{Com}_{mpk}(M; R), \mathbb{H}\mathsf{Open}_{mpk}(M; R)) = 1$ holds with all but negligible probability.

We now turn to the novel "soft algorithms". The *soft-commit* algorithm $\mathbb{S}\mathsf{Com}_{mpk}(; R)$ produces a *soft-commitment* string $C$ (to no message in particular!). We will sometimes write $C \leftarrow \mathbb{S}\mathsf{Com}_{mpk}()$. The *soft-opening* algorithm $\mathbb{S}\mathsf{Open}_{mpk}(M, \mathsf{flag}; R)$, where $M \in \mathcal{M}$ and $\mathsf{flag} \in \{\mathbb{H}, \mathbb{S}\}$ now produces a *soft-decommitment* $\tau$ to $M$, which should supposedly convey information that "if the commitment produced using $R$ can be hard-opened at all, then it would open to $M$". A bit more precisely (but see more below), if $\mathsf{flag} = \mathbb{H}$, then $\tau$ is supposed to "correspond" to the hard-commitment $C = \mathbb{H}\mathsf{Com}_{mpk}(M; R)$, and if $\mathsf{flag} = \mathbb{S}$, then $\tau$ is a fake soft-decommitment "corresponding" to the soft-commitment $C = \mathbb{S}\mathsf{Com}_{mpk}(; R)$. Either one of these cases is verified using the *soft-verification* algorithm $\mathbb{S}\mathsf{Ver}_{mpk}(M, C, \tau)$, which outputs 1 if it thinks that $C$ could potentially be hard-opened to $M$ in the future (which, intuitively, should be the case only when $\tau$ was produced from a hard-commitment). Specifically, we require that for all $M \in \mathcal{M}$, $\mathbb{S}\mathsf{Ver}_{mpk}(M, \mathbb{H}\mathsf{Com}_{mpk}(M; R), \mathbb{S}\mathsf{Open}_{mpk}(M, \mathbb{H}; R)) = 1$ holds with all but negligible probability, and similarly $\mathbb{S}\mathsf{Ver}_{mpk}(M, \mathbb{S}\mathsf{Com}_{mpk}(; R), \mathbb{S}\mathsf{Open}_{mpk}(M, \mathbb{S}; R)) = 1$ holds with all but negligible probability.

We notice that in many cases (including all our constructions) the soft-decommitment $\tau$ to a hard-commitment $C$

[6]Perhaps partially explained by the fact that mercurial commitment are trapdoor commitments with several very special properties (see Section 2.3). Correspondingly, in our main construction we will need "hard" languages also satisfying some special properties. Somehow remarkably, though, these extra properties have more or less led us to trapdoor commitments themselves! See Section 4.

will consist of some proper part of the hard-decommitment $\pi$, and, correspondingly, the soft-verification algorithm will perform a proper subset of the tests performed by the hard-verification algorithm. For a lack of better name, we call such natural mercurial commitments *proper*.

SECURITY. The binding property of plain mercurial commitments consists of two requirements, stating that a valid hard- or soft-opening of $C$ to some $M$ implies that $C$ can not be then hard-opened to any other message $M' \neq M$:

- **Mercurial Binding:** Having the knowledge of $mpk$, it is computationally hard for the PPT adversary $\mathcal{A}$ to come up with $C, M, \pi, M', \pi'$ (resp. $C, M, \tau, M', \pi'$) such that $\pi$ (respectively, $\tau$) is a valid hard- (respectively soft-) decommitment of $C$ to $M$ and $\pi'$ is a valid hard-decommitment of $C$ to $M'$, but $M \neq M'$ (such a tuple is said to cause a *hard* (respectively *soft*) *collision*). That is, $\mathcal{A}$ cannot find a value $C$ which it can hard- or soft-open in one way and then hard-open in a different way.

We remark that for proper mercurial commitments it suffices to prove that no soft collisions can be found.

As for the analog of the hiding property, we require that not only hard-commitments to some $M$ look indistinguishable from soft-commitments (to "nothing"), but this continues to hold even if they are both soft-opened to $M$ (notice that by the mercurial binding property, the hard-commitment to $M$ cannot be soft-opened to anything other than $M$).

- **Mercurial Hiding:** No PPT adversary (who knows $mpk$) can find a message $M \in \mathcal{M}$ for which it can distinguish a random "real" hard-commitment/soft-decommitment tuple $(M, \mathbb{H}\mathsf{Com}_{mpk}(M; R), \mathbb{S}\mathsf{Open}_{mpk}(M, \mathbb{H}; R))$ from a random "fake" soft-commitment/soft-decommitment tuple $(M, \mathbb{S}\mathsf{Com}_{mpk}(; R), \mathbb{S}\mathsf{Open}_{pk}(M, \mathbb{S}; R))$.

(TRAPDOOR) MERCURIAL COMMITMENTS. Such commitment schemes consist of ten efficient algorithms: $\mathcal{C} = (\mathsf{TrMCom\text{-}Gen}, \mathbb{H}\mathsf{Com}, \mathbb{H}\mathsf{Open}, \mathbb{H}\mathsf{Ver}, \mathbb{S}\mathsf{Com}, \mathbb{S}\mathsf{Open}, \mathbb{S}\mathsf{Ver}, \mathsf{MFake}, \mathbb{H}\mathsf{Equiv}, \mathbb{S}\mathsf{Equiv})$. The generation algorithm $\mathsf{TrMCom\text{-}Gen}(1^k)$, where $k$ is the security parameter, outputs a public mercurial commitment key $mpk$ and and a secret mercurial *trapdoor key* $msk$. Once $mpk$ is fixed, the meaning of $\mathbb{H}\mathsf{Com}$, $\mathbb{H}\mathsf{Open}$, $\mathbb{H}\mathsf{Ver}$, $\mathbb{S}\mathsf{Com}$, $\mathbb{S}\mathsf{Open}$ and $\mathbb{S}\mathsf{Ver}$ is exactly the same as for plain mercurial commitments. In particular, we will require that these algorithms satisfy the usual mercurial hiding and binding properties of the plain mercurial commitment schemes.

The trapdoor key $msk$ is used in the algorithms $\mathsf{MFake}$, $\mathbb{H}\mathsf{Equiv}$ and $\mathbb{S}\mathsf{Equiv}$ to break the binding property of commitments. The algorithm $\mathsf{MFake}_{msk}(; R)$ is somewhat similar in spirit to the soft-commitment algorithm $\mathbb{S}\mathsf{Com}_{mpk}$ and produces "fake" commitment $C$, initially not associated to any message $M$. The meaning of the other two algorithms $\mathbb{H}\mathsf{Equiv}_{msk}(M; R)$ and $\mathbb{S}\mathsf{Equiv}_{msk}(m; R)$ is also similar to that of the corresponding algorithms $\mathbb{H}\mathsf{Open}_{mpk}$, $\mathbb{S}\mathsf{Open}_{mpk}$, except they *always* operate on the fake commitments $C$ not really associated to any message. Specifically, $\mathbb{H}\mathsf{Equiv}(M; R)$ produces a supposedly valid hard-opening $\pi$ (called *hard-fake*) of the fake commitment $C = \mathsf{MFake}(; R)$ to $M$, while $\mathbb{S}\mathsf{Equiv}(M; R)$ produces a supposedly valid soft-opening $\tau$ (called *soft-fake*) of the fake commitment $C = \mathsf{MFake}(; R)$ to $M$. In particular, we require that for all $M \in \mathcal{M}$, $\mathbb{H}\mathsf{Ver}_{mpk}(M, \mathsf{MFake}_{mpk}(; R), \mathbb{H}\mathsf{Equiv}_{mpk}(M; R)) = 1$ holds with all but negligible probability, and similarly $\mathbb{S}\mathsf{Ver}_{mpk}(M, \mathsf{MFake}_{mpk}(; R), \mathbb{S}\mathsf{Equiv}_{mpk}(M; R)) = 1$ holds with all but negligible probability. While the ability to soft-fake such bogus commitments is consistent with the previous ability of soft-opening, the ability to hard-fake them certainly contradicts the binding property that we had, and this is exactly the function of the trapdoor key $msk$!

Somewhat similar to the equivocation property of trapdoor commitments, we require that trapdoor mercurial commitments satisfy three related equivocation conditions. In each of them we say that no efficient distinguisher $\mathcal{A}$ can non-negligibly tell apart the corresponding "real" from the corresponding "ideal" game, *even if it is given the trapdoor key $msk$ at the beginning of each real or ideal game*. In the following, the value $R$ is always random.

- **HH Equivocation:** The real game consists of $\mathcal{A}$ choosing $M \in \mathcal{M}$ and getting back $(M, \mathbb{H}\mathsf{Com}_{mpk}(M; R), \mathbb{H}\mathsf{Open}_{mpk}(M; R))$; while the ideal game consists of $\mathcal{A}$ choosing $M \in \mathcal{M}$ and getting back $(M, \mathsf{MFake}_{msk}(; R), \mathbb{H}\mathsf{Equiv}_{msk}(M; R))$.

- **HS Equivocation:** The real game consists of $\mathcal{A}$ choosing $M \in \mathcal{M}$ and getting back $(M, \mathbb{H}\mathsf{Com}_{mpk}(M;R)$, $\mathbb{S}\mathsf{Open}_{mpk}(M,\mathbb{S};R))$; while the ideal game consists of $\mathcal{A}$ choosing $M \in \mathcal{M}$ and getting back $(M, \mathsf{MFake}_{msk}(;R)$, $\mathbb{S}\mathsf{Equiv}_{msk}(M;R))$.
- **SS Equivocation:** The real game consists of $\mathcal{A}$ getting the value $C = \mathbb{S}\mathsf{Com}_{mpk}(;R)$, then choosing $M \in \mathcal{M}$, and finally getting the value $\mathbb{S}\mathsf{Open}_{mpk}(M,\mathbb{S};R)$; while ideal game consists of $\mathcal{A}$ getting the value $C = \mathsf{MFake}_{mpk}(;R)$, then choosing $M \in \mathcal{M}$, and finally getting the value $\mathbb{S}\mathsf{Equiv}_{mpk}(M;R)$.

Notice that similar-looking SH condition does not make sense in the real life (due to mercurial binding). Next, HS and SS Equivocations easily imply the Mercurial Hiding property, so it does not need to be checked. Also, for proper mercurial commitments it is easy to see that HH equivocation implies HS equivocation, so it is enough to check only HH and SS Equivocations.

RELATION TO THE ORIGINAL DEFINITION IN [8]. The main difference from [8] is in the equivocation property, which is considerably simpler to state and understand in our case. Moreover, it is also *stronger* than the definition of [8]. Essentially, the latter definition consists of playing an arbitrary composition of HH, HS and SS Equivocation games either in the real, or in the ideal world,[7] but where the distinguisher $\mathcal{A}$ is *not given the trapdoor key* $msk$. In this scenario the usual hybrid argument does not work (since $\mathcal{A}$ cannot simulate stuff in the ideal world by himself), so one cannot reduce the composed game to the one of the three atomic HH, SE or SS games. As a result, one has to build a full-fledged simulator, and formally argue that it fools the distinguisher. In contrast, in our scenario the hybrid argument easily works, so the security of our 3 atomic games easily implies the security of the composed game *even if the distinguisher knows* $msk$.

KNOWN CONSTRUCTIONS. Chase et al. [8] gave several elegant constructions of (trapdoor) mercurial commitments from the following assumptions:

- *Non-interactive zero-knowledge proofs (*NIZK*) for* NP *[2, 14] and unconditionally-binding commitment schemes*. However, this construction is mainly of theoretical interest, since all known NIZK constructions (especially for all of NP) are extremely inefficient. Interestingly, it also does *not* satisfy our stronger definition. However, we believe that this is not a problem, since in the sequel we will provide more general constructions (from one-way functions) which satisfy our stronger definition and are still more efficient than this construction.[8]
- *Claw-free permutations [21].* This construction give only bit mercurial commitment, and will be a special case of our first general construction from bit trapdoor commitments.
- *Discrete log.* This is a "distillation" of the original construction implicitly used in [27]; it supports long messages and is pretty efficient. It will be a special case of our second construction when used with the corresponding discrete-log based bit trapdoor commitment.
- *Factoring.* This is a new construction which supports long messages and is relatively efficient. It will be a special case of our second construction when used with the corresponding factoring-based bit trapdoor commitment.

[7]There is one other, more syntactic strengthening that we had to make in order to simplify the definition. Namely, in the more general definition of [8] one could have syntactically unrelated real and ideal experiments for generating $mpk$, so it did not make sense to give $msk$ to $\mathcal{A}$ in the real game. In contrast, we insist that the public key generation even in the real world can be carried by generating both the public and the trapdoor key. While slightly more restrictive, since (1) all our efficient constructions satisfy this restricted notion of key generation and (2) it considerably simplifies (and also *strengthens*) the definition, we feel it is very justified.

[8]We remark, however, that the NIZK construction has the advantage of being in the common *random string* (CRS) model, as opposed to the common *parameter* model that we use. This means that its public key is computationally indistinguishable from random. We notice that by using (non-interactive) trapdoor commitments *in the CRS model*, — which can be built from one-way *permutations* (see [7]) or specific number-theoretic assumptions (such as discrete log), — our constructions will also give CRS-based mercurial commitments (even satisfying our stronger definition). The recent independent work of Catalano and Visconti [7] shows how to base mercurial commitments on one-way *functions* in the CRS model, but only using the weaker definition of [8]. It is open how to satisfy our stronger definition in the CRS model assuming only one-way functions.

IMPLICATIONS TO (TRAPDOOR) COMMITMENTS. It is simple to see that by "ignoring" all the "soft" algorithms of a secure plain/trapdoor commitment scheme, we immediately get a plain/trapdoor regular commitment scheme. (Concentrating, for example, on a slightly more complicated "trapdoor case", $\mathbb{H}\mathsf{Com}$ plays the role of $\mathsf{Com}$, $\mathbb{H}\mathsf{Open}$ — of $\mathsf{Open}$, $\mathbb{H}\mathsf{Ver}$ — of $\mathsf{Ver}$, $\mathsf{MFake}$ — of $\mathsf{Fake}$, and $\mathbb{H}\mathsf{Equiv}$ — of $\mathsf{Equiv}$.) In the following, we show two simple constructions proving that the converse of this statement is true as well.

## 3 General Construction from (Trapdoor) Bit Commitments

As advocated in the introduction, we will first consider the construction of plain mercurial bit commitments from regular bit commitments, and then argue that the same construction extends to the trapdoor case as well.

BUILDING PLAIN MERCURIAL COMMITMENTS. Assume $\mathcal{C} = (\mathsf{Com\text{-}Gen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Ver})$ is a regular bit commitment scheme. Define plain mercurial commitment $\mathcal{C}' = (\mathsf{MCom\text{-}Gen}, \mathbb{H}\mathsf{Com}, \mathbb{H}\mathsf{Open}, \mathbb{H}\mathsf{Ver}, \mathbb{S}\mathsf{Com}, \mathbb{S}\mathsf{Open}, \mathbb{S}\mathsf{Ver})$ for a bit $b \in \{0,1\}$ as follows (we set $\mathsf{MCom\text{-}Gen} = \mathsf{Com\text{-}Gen}$ and let $pk$ be the corresponding public key):

- $\mathbb{H}\mathsf{Com}_{pk}(b; (r_0, r_1))$: output $(c_0, c_1) = (\mathsf{Com}_{pk}(b; r_0), \mathsf{Com}_{pk}(1-b; r_1))$. Notice, commitment to $0$ changes its place from left to right depending on $b$.
- $\mathbb{H}\mathsf{Open}_{pk}(b; (r_0, r_1))$: output $(d_0, d_1) = (\mathsf{Open}(b; r_0), \mathsf{Open}(1-b; r_1))$.
- $\mathbb{H}\mathsf{Ver}_{pk}(b, (c_0, c_1), (d_0, d_1))$: accept if and only if $\mathsf{Ver}_{pk}(b, c_0, d_0) = \mathsf{Ver}_{pk}(1-b, c_1, d_1) = 1$.
- $\mathbb{S}\mathsf{Com}_{pk}(; (r_0, r_1))$: output $(c_0, c_1) = (\mathsf{Com}_{pk}(0; r_0), \mathsf{Com}_{pk}(0; r_1))$.
- $\mathbb{S}\mathsf{Open}_{pk}(b, \mathsf{flag}; (r_0, r_1))$: irrespective of $\mathsf{flag} \in \{\mathbb{H}, \mathbb{S}\}$, output $d = \mathsf{Open}(0; r_b)$.
- $\mathbb{S}\mathsf{Ver}_{pk}(b, (c_0, c_1), d)$: accept if and only if $\mathsf{Ver}_{pk}(0, c_b, d) = 1$.

The correctness of the scheme is obvious. Intuitively, mercurial commitment to $b = 0$ looks $(0,1)$, to $1$ — $(1,0)$, and the fake — $(0,0)$. Since the soft-opening of the hard commitment only opens the corresponding left or right $0$, the fake commitment can indeed be soft-opened in both way, by honestly opening the appropriate left of right $0$. On the other hand, seeing a hard-opening of some commitment $C = (c_0, c_1)$ (to some bit $b$) opens to $1$ one of the two regular commitments, while the subsequent soft-opening of $C$ to $(1-b)$ would then open this regular commitment to $0$, which contradicts binding. Below, we formalize this is a straightforward manner.

*Mercurial Binding.* Since the mercurial commitment is proper, we only need to rule out soft collisions. For that, assume the attacker can find a soft collision. By symmetry, let us assume that $1$ is the softly-opened message, and $0$ is the hardly-opened one). So we denote this collision by $C = ((c_0, c_1), d_0, d_1, d_1')$ where $\mathsf{Ver}(0, c_0, d_0) = \mathsf{Ver}(1, c_1, d_1) = \mathsf{Ver}(0, c_1, d_1') = 1$. But then $c_1$ can be opened to both $0$ and $1$, a contradiction to the binding property of $\mathcal{C}$.

*Mercurial Hiding.* Assume first $b = 0$. Then, the "real" hard-commitment/soft-decommitment tuple $(\mathbb{H}\mathsf{Com}(0; (r_0, r_1)), \mathbb{S}\mathsf{Open}(0, \mathbb{H}; (r_0, r_1))$ looks like $(\mathsf{Com}(0; r_0), \mathsf{Com}(1; r_1), \mathsf{Open}(0; r_0))$, while the corresponding "fake" tuple $(\mathsf{Fake}(; (r_0, r_1)), \mathbb{S}\mathsf{Open}(0, \mathbb{S}; (r_0, r_1))$ looks like $(\mathsf{Com}(0; r_0), \mathsf{Com}(0; r_1), \mathsf{Open}(0; r_0))$. Clearly, such distribution are indistinguishable if $\mathsf{Com}(0)$ cannot be distinguished from $\mathsf{Com}(1)$, which follows from the hiding property of $\mathcal{C}$. A similar argument holds for $b = 1$ as well.

TRAPDOOR CASE. The extension to the trapdoor case is simple as well. We now have additional algorithms $\mathsf{Fake}$ and $\mathsf{Equiv}$ for trapdoor commitments, and need to build the corresponding algorithms $\mathsf{MFake}$, $\mathbb{H}\mathsf{Equiv}$ and $\mathbb{S}\mathsf{Equiv}$ for mercurial commitments.

- $\mathsf{MFake}_{sk}(; (r_0, r_1))$: output $(\mathsf{Fake}_{sk}(; r_0), \mathsf{Fake}_{sk}(; r_1))$.
- $\mathbb{H}\mathsf{Equiv}_{sk}(b; (r_0, r_1))$: output $(\mathsf{Equiv}_{sk}(b; r_0), \mathsf{Equiv}_{sk}(1-b; r_1))$.
- $\mathbb{S}\mathsf{Equiv}_{sk}(b; (r_0, r_1))$: output $\mathsf{Equiv}_{sk}(0; r_b)$.

Correctness is obvious from definition. As for hiding, we only need to argue HH and SS Equivocability (since this is a proper mercurial commitment). Both are simple corollaries of the regular Equivocation properties of trapdoor commitments.

*HH Equivocation.* Let us assume $b = 0$, since $b = 1$ is symmetric. Then $(\mathbb{H}\mathsf{Com}(0; (r_0, r_1)), \mathbb{H}\mathsf{Open}(0; (r_0, r_1)))$ is equal to $D_{real} = (\mathsf{Com}(0; r_0), \mathsf{Com}(1; r_1), \mathsf{Open}(0; r_0), \mathsf{Open}(1; r_1))$, while $(\mathsf{MFake}(; (r_0, r_1)), \mathbb{H}\mathsf{Equiv}(0; (r_0, r_1)))$ is equal to $D_{ideal} = (\mathsf{Fake}(; r_0), \mathsf{Fake}(; r_1), \mathsf{Equiv}(0; r_0), \mathsf{Equiv}(1; r_1))$. Since $r_0$ and $r_1$ are independent, this amount to two independent applications of the regular Equivocation property to bits $0$ and $1$, respectively. Notice, though, already for this simple hybrid argument *we are using the fact that the attacker knows the trapdoor key $sk$*! To be precise, we must first consider a hybrid distribution $D_{hyb} = (\mathsf{Fake}(; r_0), \mathsf{Com}(1; r_1), \mathsf{Equiv}(0; r_0), \mathsf{Open}(1; r_1))$, and then show $D_{real} \approx D_{hyb}$ (here we only need $pk$ to sample $(\mathsf{Com}(1; r_1), \mathsf{Open}(1; r_1))$) and $D_{hyb} \approx D_{ideal}$ (here we *need $sk$* to sample $(\mathsf{Fake}(; r_0), \mathsf{Equiv}(0; r_0))$).

*SS Equivocation.* In the real experiment, the attacker is first getting $(\mathsf{Com}(0; r_0), \mathsf{Com}(0; r_1))$, then he has to choose a bit $b$, after which he gets $\mathsf{Open}(0; r_b)$. In the ideal game, the attacker is getting $(\mathsf{Fake}(; r_0), \mathsf{Fake}(; r_1))$, then he has to choose a bit $b$, after which he gets $\mathsf{Equiv}(0; r_b)$. By symmetry, the choice of $b$ does not matter here, so we can assume $b = 0$, so it suffices to argue $(\mathsf{Com}(0; r_0), \mathsf{Com}(0; r_1), \mathsf{Open}(0; r_0)) \approx (\mathsf{Fake}(; r_0), \mathsf{Fake}(; r_1), \mathsf{Equiv}(0; r_0))$. Once again, this follow by the hybrid argument, by considering an intermediate distribution $(\mathsf{Fake}(; r_0), \mathsf{Com}(; r_1), \mathsf{Equiv}(0; r_0))$ and using the fact that in the second hybrid the attacker can compute $(\mathsf{Fake}(; r_0),\ \mathsf{Open}(0; r_0))$.

COMPARISON TO [8]. The above construction is a very simple generalization of the one in [8], who used the following family of trapdoor bit commitments [26] obtained from any family of claw-free permutations [21] $(f_0, f_1)$. Informally, recall that these are pairs of permutations where one cannot find a "claw" $(r_0, r_1)$ satisfying $f_0(r_0) = f_1(r_1)$; also it is assumed that there exists a trapdoor $f_0^{-1}$ allowing one to invert $f_0$ (in our application, we will not need a similar trapdoor for $f_1$). Now, to trapdoor commit to a bit $b$ we can sample $f_b(r_b)$ (decommitment is $r_b$), while the knowledge of the trapdoor $f_0^{-1}$ provides easy fake pairs: the fake commitment $c = f_1(r_1)$ (for random $r_1$) can be opened to $0$ by giving $r_0 = f_0^{-1}(c)$), and to $1$ — by giving $r_1$.

We remark, though, that the equivocability proof of our extension is indeed considerably shorter, — which is what it should be for such a simple construction! — than the corresponding proof [8]. Also, our construction implies mercurial commitments from other bit commitments which are not necessarily induced by claw-free permutations, such as the general construction of [15] from any $\Sigma$-protocol for a hard language, the factoring construction of [31], the Paillier construction of [5] or the ad hoc $(g^{r_0}, h^{r_1})$-construction mentioned in Section 2.2 (and, of course, the one-way function construction from Theorem 4).

## 4 Efficient Construction from (Trapdoor) Bit Commitments with $\Sigma$-Protocols

The problem with the previous generic construction is the fact that it only allows one to commit to one bit. Of course, we can always commit to many bits by following the "bit-by-bit" approach, but this is inefficient. Alternatively, we can try to utilize a multi-bit plain/trapdoor commitment scheme in the previous construction, but it is easy to see that the resulting length of the commitment will be linearly proportional to the number of *messages* that we want to commit to. This essentially means that setting this number to 2 — as we did in Section 3 — and doing the bit-by-bit composition is the best we can do if we try to extend the previous approach.

Instead, in this section we present our main construction which will directly transforms a plain/trapdoor bit commitment $\mathcal{C}$ into an efficient and (potentially) multi-bit plain/mercurial commitment $\mathcal{C}'$. However, we will need to assume an extra property from $\mathcal{C}$: there exists an efficient $\Sigma$-protocol $\Pi$ proving that one knows a witness $d$ that a given commitment $c$ can be opened to $0$. In this case, the message space of $\mathcal{C}'$ will be the challenge space of the corresponding $\Sigma$-protocol. Thus, if $\Pi$ will be challenge-efficient, we would get a direct, large-message mercurial commitment $\mathcal{C}'$.

CONSTRUCTION. Assume $\mathcal{C} = (\mathsf{Com\text{-}Gen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Ver})$ is a regular bit commitment scheme such that there exists a $\Sigma$-protocol $\Pi = (\mathsf{Start}, \mathsf{Finish}, \mathsf{Extract}, \mathsf{Simul})$ for the relation (family) $\mathcal{R}_{pk} = \{(c, d) \mid \mathsf{Ver}_{pk}(0, c, d) = 1\}$. Recall, this means that the verifier only gets a commitment $c$, and the prover also gets, as a witness, a valid opening $d$ of $c$ to $0$. Also, assume $\mathcal{M}$ is the challenge space for $\Pi$.

We then define plain mercurial commitment $\mathcal{C}' = (\mathsf{MCom\text{-}Gen}, \mathbb{H}\mathsf{Com}, \mathbb{H}\mathsf{Open}, \mathbb{H}\mathsf{Ver}, \mathbb{S}\mathsf{Com}, \mathbb{S}\mathsf{Open}, \mathbb{S}\mathsf{Ver})$ for message space $\mathcal{M}$ as follows (we set $\mathsf{MCom\text{-}Gen} = \mathsf{Com\text{-}Gen}$ and let $pk$ be the corresponding public key):

- $\mathbb{H}\mathsf{Com}_{pk}(m; (r_s, r_1))$: let $c_1 = \mathsf{Com}_{pk}(1; r_1)$ be a commitment to 1, and $(a_1, z_1) = \mathsf{Simul}_{pk}(c_1, m; r_s)$ be a fake first and last messages of $\Pi$ which (here incorrectly) claim that $c_1$ is a commitment to $0$ on challenge $m$. Output $(c_1, a_1)$.
- $\mathbb{H}\mathsf{Open}_{pk}(m; (r_s, r_1))$: let $c_1 = \mathsf{Com}_{pk}(1; r_1)$ and $(a_1, z_1) = \mathsf{Simul}_{pk}(c_1, m; r_s)$ be as before. Set $d_1 = \mathsf{Open}_{pk}(1; r_1)$ and output $(d_1, z_1)$.
- $\mathbb{H}\mathsf{Ver}_{pk}(m, (c_1, a_1), (d_1, z_1))$: accept if and only if $\mathsf{Ver}_{pk}(1, c_1, d_1) = 1$ ($d_1$ is correct decommitment to 1) and $\mathsf{Check}_{pk}(c_1, a_1, m, z_1) = 1$ (the fake transcript on challenge $m$ that $c_1$ is a commitment to $0$ looks good).
- $\mathbb{S}\mathsf{Com}_{pk}(; (r_p, r_0))$: let $c_0 = \mathsf{Com}_{pk}(0; r_0)$ be a commitment to 0, $d_0 = \mathsf{Open}_{pk}(0; r_0)$ be the corresponding opening, and $a_0 = \mathsf{Start}_{pk}(c_0, d_0; r_p)$ be a real first messages of $\Pi$ which (correctly!) claims that $c_0$ is a commitment to 0.[9] Output $(c_0, a_0)$.
- $\mathbb{S}\mathsf{Open}_{pk}(m, \mathbb{H}; (r_s, r_1))$: let $c_1 = \mathsf{Com}_{pk}(1; r_1)$ and $(a_1, z_1) = \mathsf{Simul}_{pk}(c_1, m; r_s)$ be the fake transcript on challenge $m$ that $c_1$ is a commitment to 0. Output $z_1$.
- $\mathbb{S}\mathsf{Open}_{pk}(m, \mathbb{S}; (r_p, r_0))$: let $c_0 = \mathsf{Com}_{pk}(0; r_0)$, $d_0 = \mathsf{Open}_{pk}(0; r_0)$, $a_0 = \mathsf{Start}_{pk}(c_0, d_0; r_p)$, and $z_0 = \mathsf{Finish}_{pk}(c_0, d_0, m; r_p)$ be the correct last flow to challenge $m$. Output $z_0$.
- $\mathbb{S}\mathsf{Ver}_{pk}(m, (c, a), z)$: accept if and only $\mathsf{Check}_{pk}(c_b, a, m, z) = 1$ (the transcript $(a, m, z)$ stating that $c$ is a commitment to 0 is correct).

Intuitively, the honest hard-committer is supposed to send a commitment $c$ to 1, but *fake* the transcript that he in fact committed to 0. On the other hand, a lying soft-committer can simply send a commitment $c$ to 0, and now can (*honestly*!) respond to any challenge/message $m$ that he gets subsequently, which allows him to soft-open the first flow to any message $m$.[10] The binding security of this scheme comes from the fact that a hard-opening of $c$ to 1, coupled with two soft-opening of the first flow $a$, must enable one to extract a legal witness, which is the hard-opening of $c$ to 0, thus contradicting the binding of $\mathcal{C}$. Similarly, the hiding property of the commitment coupled with the zero-knowledge property of $\Sigma$-protocols imply that, without the hard-opening of $c$ (which will tell if $c$ is a commitment to 0 or 1), the real and fake behavior cannot be told apart. More formally,

*Mercurial Binding.* Since our commitment is proper, we only need to rule our soft collisions. This means that the attacker can find a commitment value $(c, a)$, a decommitment $d_1$ proving that $c$ is a commitment to 1, two messages $m \neq m'$, and two valid responses $z$ and $z'$ claiming that $c$ is a commitment to 0. By the special soundness of the $\Sigma$-protocol, we get that $\mathsf{Extract}(c, a, m, z, m', z')$ must be equal to a valid decommitment $d_0$ of $c$ to 0. But then we found a way to open $c$ to both 0 and 1 (via $d_0$ and $d_1$), contradicting the binding property of $\mathcal{C}$.

*Mercurial Hiding.* Take any message/challenge $m$. Then, the "real" hard-commitment/soft-decommitment tuple for $m$ looks like is given by three values $(c = \mathsf{Com}(1; r_1), (a, z) = \mathsf{Simul}(c, m; r_s))$. Since our commitment is hiding, and $\mathsf{Simul}(c, m)$ is publicly computable, we get that the above distribution is indistinguishable from $(c = \mathsf{Com}(0; r_0), (a, z) = \mathsf{Simul}(c, m; r_s))$. Now, since $c$ has a proper witness $d_0 = \mathsf{Open}(0; r_0)$, the special HVZK property of $\Pi$ states that the distribution on $(a, z)$ looks indistinguishable than the one obtained by a running

[9]Notice, here the prover actually knows the value $r_0$, and not just $d_0$. So for efficiency reasons we might consider auxiliary-input $\Sigma$-protocols where $P$'s witness is actually $r_0$ itself. We will return to this point later.

[10]Is might appear peculiar that we require an honest party to cook-up a fake proof in order to succeed, while having a dishonest party perform such a proof correctly! Here, however, the primitive we build legally allows a dishonest party to look "slightly like an honest party". So the we force the honest party to do something slightly bad which might be "matched" by a good action of a dishonest party.

a real protocol on input $c$, witness $d_0$ and challenge $m$. But this means that the above distribution is indistinguishable from $(c = \mathsf{Com}(0; r_0), a = \mathsf{Start}(c, d_0; r_p), z = \mathsf{Finish}(c, d_0, m; r_p))$, which is exactly the triple corresponding to the "fake" soft-commitment/soft-decommitment procedures.

TRAPDOOR CASE. Recall, we now have additional algorithms $\mathsf{Fake}$ and $\mathsf{Equiv}$ for trapdoor commitments, and need to build the corresponding algorithms $\mathsf{MFake}$, $\mathbb{H}\mathsf{Equiv}$ and $\mathbb{S}\mathsf{Equiv}$ for mercurial commitments. As a new technical property about the $\Sigma$-protocol, however, we will have to assume that $\Pi = \Pi_{pk}$ is strongly hiding w.r.t. a particular parameter generation procedure $\mathcal{P}$ (see Section 2.1). The parameter generation procedure we will need generates random keys $(pk, sk) \leftarrow \mathsf{Com\text{-}Gen}(1^k)$, picks a random $r$, computes $c = \mathsf{Fake}_{sk}(; r)$, $d_0 = \mathsf{Equiv}_{sk}(0; r)$, $d_1 = \mathsf{Equiv}_{sk}(1; r)$, and sets the side information to $(sk, d_1)$, the input to be $c$ , and the witness to be $d_0$. As explained in Section 2.1, this is more of a technicality which seems to be always satisfied in any non-pathological scenario arising in practice. We call this property (*), and can now describe the claimed extension.

- $\mathsf{MFake}_{sk}(; (r_p, r))$: let $c = \mathsf{Fake}_{sk}(; r)$ be a fake commitment, $d_0 = \mathsf{Equiv}_{sk}(0; r)$ be its fake opening to $0$, and $a_0 = \mathsf{Start}_{pk}(c, d_0; r_p)$ be a correct first flow of the $\Sigma$-protocol. Output $(c, a_0)$.
- $\mathbb{H}\mathsf{Equiv}_{sk}(m; (r_p, r))$: let $c = \mathsf{Fake}_{sk}(; r)$, $d_0 = \mathsf{Equiv}_{sk}(0; r)$, and $a_0 = \mathsf{Start}_{pk}(c, d_0; r_p)$ be as before. Compute the fake opening $d_1 = \mathsf{Equiv}_{sk}(1; r)$ of $c$ to 1, and the correct last message $z_0 = \mathsf{Finish}_{pk}(c, d_0, m; r_p)$. Output $(d_1, z_0)$.
- $\mathbb{S}\mathsf{Equiv}_{sk}(m; (r_p, r))$: let $c = \mathsf{Fake}_{sk}(; r)$, $d_0 = \mathsf{Equiv}_{sk}(0; r)$, and $a_0 = \mathsf{Start}_{pk}(c, d_0; r_p)$ be as before. Compute the correct last message $z_0 = \mathsf{Finish}_{pk}(c, d_0, m; r_p)$ and output $z_0$.

Correctness is obvious from definition. As for hiding, we only need to argue HH and SS Equivocability (since this is a proper mercurial commitment).

*HH Equivocation.* Take any message $m$. Then $(\mathbb{H}\mathsf{Com}(m; (r_s, r_1)), \mathbb{H}\mathsf{Open}(m; (r_s, r_1)))$ is equal to $D_{real} = (c_1, d_1, a_1, z_1)$, where $c_1 = \mathsf{Com}(1; r_1)$, $d_1 = \mathsf{Open}(1; r_1)$, and $(a_1, z_1) = \mathsf{Simul}(c_1, m; r_s))$. Since $\mathsf{Simul}(c_1, m)$ is a public transformation, the Equivocability of $\mathcal{C}$ implies that the above distribution is indistinguishable from $(c = \mathsf{Fake}(; r), d_1 = \mathsf{Equiv}(1; r), a_1, z_1)$, where $(a_1, z_1) = \mathsf{Simul}(c, m; r_s)$. We are almost done, except we need to replace the above $(a_1, z_1)$ by $(a_0, z_0)$ obtained by running an honest execution of $\Pi$ with witness $d_0 = \mathsf{Equiv}(0; r)$. This is almost exactly the HVZK property, except we formally need to use the strong hiding property (*) described above. Indeed, in addition to the input $c$ and the public parameter $pk$, which are allowed in the usual HVZK property, here the distinguisher also knows two extra pieces of information: the trapdoor key $sk$ (given to him at the beginning of the game) and the fake decommitment $d_1 = \mathsf{Equiv}(1; r)$. This is why we needed to assume that this extra information does not violate the HVZK property.

*SS Equivocation.* In the real soft-commit/soft-open experiment, the distinguisher (who knows $sk$) is first getting $c_0 = \mathsf{Com}(0; r_0)$ and the correct first flow of the $\Sigma$-protocol showing that $c_0$ is indeed a commitment to 0 (using witness $d_0 = \mathsf{Open}(0; r_0)$). He then chooses a message $m$, and gets a correct third flow to message $m$. To put differently, he simply plays the role of (malicious) verifier in the honest run of the $\Sigma$-protocol on pair $(c_0, d_0)$. Notice that the distinguisher's view can be perfectly simulated using some public probabilistic procedure $A_{sk}(c_0, d_0)$. Using the equivocation property of $\mathcal{C}$, the resulting distribution should be indistinguishable from $A_{sk}(c, d_0)$, where $c = \mathsf{Fake}(; r)$ and $d_0 = \mathsf{Equiv}(0; r)$. But, once again, it is easy to see that this view is exactly what the attacker gets in the ideal soft-commit/soft-open experiment.

GENERALIZATION. We already noticed in Footnote 9 that in the above definition of soft-commitment, the Prover actually knows the entire randomness $r_0$ and not just a witness $d_0 = \mathsf{Open}(0; r_0)$. This, of course, is of any value only in a very few schemes where $r_0 \neq d_0$. However, it will come up in one of our examples (see Section 4.1). To accommodate this extension, we can consider $\Sigma$-protocol's where the prover needs all of $r_0$ for the completeness of the protocol (special soundness is still only for $d_0$). For plain mercurial commitments, this is all we need to change. For the trapdoor variant, however, we will need an extra property from our trapdoor commitment scheme in regards to equivocation. Namely, in the fake commitment algorithm we need to be able to equivocate $c = \mathsf{Fake}(; r)$ to $0$

by obtaining not only a good looking value $d_0$, but the entire randomness $r_0$. To put differently, we will need to assume a trapdoor commitment which allows one to equivocate not only the decommitment values, but the entire randomness. Once this is ensured, we can easily support auxiliary input $\Sigma$-protocols.

## 4.1 Examples

Below we briefly give several efficient instantiations of our construction, by applying it to several efficient trapdoor commitment schemes with challenge-efficient $\Sigma$-protocols. As we will see, our examples will cover *all* the previous efficient schemes, and several more, all as part of one general framework. Except for the first two examples — which are exactly the schemes from [8] — for the remaining schemes we will just briefly mention which trapdoor commitment and $\Sigma$-protocol to use, since the remaining details are obvious and not very illuminating.

DISCRETE LOG CONSTRUCTION FROM [8, 27]. We will consider the ad-hoc scheme from Section 2.2, where $\mathsf{Com}(0; r_0) = g^{r_0}$, $\mathsf{Com}(1; r_1) = h^{r_1}$, and the trapdoor $sk = \log_g h$ (here $r_0, r_1 \neq 0$). We need a $\Sigma$-protocol to prove the knowledge of $r_0 = \log_g(c)$, where $c$ is the claimed commitment to 0. Of course, a natural thing to do is to take Schnorr protocol, but this will result in a slightly different (but equally efficient) scheme than what we are after. Instead, we will use a bit less esthetic but equally effective $\Sigma$-protocol. In the first flow the prover sends a value $T = g^t$ (for random $t$), he gets challenge $m$, and responds with $z = (t - m)/r_0 \bmod q$ (which is defined since $r_0 \neq 0$). The verifier checks if $g^m c^z = T$ (indeed, $m + r_0 z = t$, as needed). It is simple to see that this is indeed a $\Sigma$-protocol for the knowledge of the discrete log, and that by plugging it into our construction we get exactly the discrete log construction from [8, 27].

We also remark what we could use a better known discrete-log commitment $\mathsf{Com}(0) = g^{r_0}$, $\mathsf{Com}(1) = hg^{r_1}$, coupled with either Schnorr $\Sigma$-protocol, or the one presented above. We will get yet another (equally efficient) solution.

FACTORING CONSTRUCTION FROM [8]. This will use the generalization of our constriction to use auxiliary inputs, as explained at the end of the previous section. To motivate the construction, though, let us start with a well-known factoring-based trapdoor bit commitment from a corresponding claw-free permutation pair: the public parameter is a random square $U$, and $\mathsf{Com}(0; r_0) = r_0^2 \bmod n$, $\mathsf{Com}(1; r_1) = Ur_1^2 \bmod n$ (the trapdoor is the square root of $U$). Here we need a $\Sigma$-protocol for the knowledge of the square root. As we mentioned in Section 2.1, using Fiat-Shamir protocol [17] is not communication- or challenge-efficient. Instead, we use the auxiliary input Ong-Schnorr protocol [29]. For that one need to know $2^\ell$-th square root of $\mathsf{Com}(0)$, so we modify $\mathsf{Com}(0; r_0) = r_0^{2^\ell} \bmod n$ (but leave $\mathsf{Com}(1; r_1) = Ur_1^2 \bmod n$). W'e notice, that although the decommitment to 0 is "only" the square root $d_0 = r_0^{2^{\ell-1}}$, and not $r_0$ itself, the fake commitment should enable us to extract (using $sk$) the $2^\ell$-th root from $c_0$, and not just a mere square root. Of course, this is easy to achieve by defining $\mathsf{Fake}(; r) = r^{2^\ell}$, and "fully opening" it to 0 by giving $r$, and to 1 — by giving $r^{2^{\ell-1}}/\sqrt{U}$. With these changes we get *precisely* the factoring construction from [8].

We also notice that by using a different claw-free permutation $(r_0^2, 4r_1^2)$ [21] defined over the so called Williams integers, we can slightly simplify the scheme and set $U = 4$.

NEW RSA-BASED CONSTRUCTION. Here we could use the RSA-based trapdoor commitment $\mathsf{Com}(0; r_0) = r_0^e \bmod n$, $\mathsf{Com}(1; r_1) = yr_1^e \bmod n$, where $y$ is a public parameter, whose $e$-th root is the trapdoor key. Here we simply need the $\Sigma$-protocol proof of knowledge of the $e$-th root, which is just the GQ protocol [23]. To have the protocol to be challenge-efficient, though, we will need to use a relatively large $e$.

ANOTHER FACTORING CONSTRUCTION. We can use the following factoring-based commitment of [31] (slightly modified for easier $\Sigma$-protocols and specialized to bits). The public key is $n = p, q$, where $p = 2p' + 1, q = 2q' + 1$ are safe primes, and all the operations are performed in the subgroup $Q_n$ of quadratic residues whose generator $g$ is also part of parameters. Notice, $|Q_n| = p'q'$. Let $C$ be a large enough constant (anything larger than $n$ will do). Then $\mathsf{Com}(0; r_0) = g^{C + r_0} \bmod n$, $\mathsf{Com}(1; r_1) = g^{r_1} \bmod n$ (here $r_0, r_1$ are random from 0 to $n$, as this is statistically close to $\varphi(n)$, which is the "true range" we are aiming for). The trapdoor is the value $|Q_n| = p'q'$. In this case the $\Sigma$-protocol we need to again the one of knowledge of discrete-log, but in the groups of unknown order. As mentioned before, such (computationally sound) protocol is given by [18, 9].

PAILLIER-BASED SCHEME. Finally, we mention another trapdoor commitment based on the hardness of finding $n$-th roots over $Z_{n^2}$ (where $n$ is the the product of two safe primes, for simplicity), which is implicit in [11].

Here the public parametrs will include a generator $g$ of the subgroup $S$ of $n$-th powers in $\mathbb{Z}^*_{n^2}$, and the $n$-th root $u$ of $g$ will be the trapdoor. Next, $\mathsf{Com}(0; r_0) = r_0^n \bmod n^2$, $\mathsf{Com}(1; r_1) = gr_1^n \bmod n^2$ (here $r_0, r_1 \in \mathbb{Z}^*_n$). This scheme is perfectly hiding and computationally binding assuming it is hard to take $n$-th root over $\mathbb{Z}_{n^2}$, and could be viewed as yet another claw-free based construction. The $\Sigma$-protocol for commitment to $0$ is simply the $\Sigma$-protocol for knowing the $n$-th root. This protocol is very similar to the GQ protocol and is formally analyzed by [10].

## 5 Conclusions

We believe that our results eludicate the notion of mercurial commitments, put them in their place on the map of cryptographic assumptions, and better explain the rational following the previous constructions of [27, 8]. We hope that mercurial commitments will find more interesting applications in the future.

## Acknowledgments

I would like to thank Leonid Reyzin for several insightful conversations about mercurial commitments, and Tal Malkin for giving a talk which stimulated this research.

## References

[1] M. Abdalla and L. Reyzin. A new forward-secure digital signature scheme. In T. Okamoto, editor, *Advances in Cryptology—ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 116–129, Kyoto, Japan, 3–7 Dec. 2000. Springer-Verlag. Full version available from the Cryptology ePrint Archive, record 2000/002, `http://eprint.iacr.org/`.

[2] Manuel Blum, Alfredo De Santis, Silvio Micali, and Guiseppe Persiano. Non-interactive zero-knowledge. *SIAM Journal of Computing*, 20(6), 1991.

[3] Joan Boyar, S. A. Kurtz, Mark W. Krentel. A Discrete Logarithm Implementation of Perfect Zero-Knowledge Blobs. In *J. of Cryptology*, 2(2):63–76, 1990.

[4] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, Oct. 1988.

[5] Dario Catalano, Rosario Gennaro, Nick Howgrave-Graham, Phong Q. Nguyen. Paillier's cryptosystem revisited. In *ACM Conference on Computer and Communications Security 2001*, pp. 206–214.

[6] Dario Catalano, Yevgeniy Dodis and Ivan Visconti. Mercurial Commitments: Minimal Assumptions and Efficient Constrcutions. In *Theory of Cryptography Conference (TCC)*, 2006.

[7] Dario Catalano, Ivan Visconti. Non-Interactive Mercurial Commitments from One-Way Functions. *ECRYPT Technical Report*, 2005.

[8] Melissa Chase, Alexander Healy, Anna Lysysanskaya, Tal Malkin and Leonid Reyzin. Mercurial Commitments with Applications to Zero-Knowledge Sets. In *Proc. of EUROCRYPT*, pp. 422–439, 2005.

[9] Ivan Damgård, Eiichiro Fujisaki. A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order. In *ASIACRYPT 2002*, pp. 125–142.

[10] Ivan Damgård, Mats Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. *Public Key Cryptography 2001*, pp. 119–136.

[11] I. Damgård and J. B. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In M. Yung, editor, *Advances in Cryptology—CRYPTO 2002*, Lecture Notes in Computer Science. Springer-Verlag, 18–22 Aug. 2002.

[12] I. B. Damgård, T. P. Pedersen, and B. Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *Journal of Cryptology*, 10(3):163–194, Summer 1997.

[13] Y. Dodis and L. Reyzin. On the power of claw-free permutations. In *Conference on Security in Communication Networks*, 2002.

[14] U. Feige, D. Lapidot, and A. Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Computing*, 29(1), 1999.

[15] U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 526–545. Springer-Verlag, 1990, 20–24 Aug. 1989.

[16] U. Feige and A. Shamir. Witness indistinguishability and witness hiding protocols. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 416–426, Baltimore, Maryland, 14–16 May 1990.

[17] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987, 11–15 Aug. 1986.

[18] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In B. S. Kaliski Jr., editor, *Advances in Cryptology—CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer-Verlag, 17–21 Aug. 1997.

[19] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[20] S. Goldwasser, S. Micali, and C. Rackoff. Knowledge complexity of interactive proofs. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, pages 291–304, Providence, Rhode Island, 6–8 May 1985.

[21] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2), 1988.

[22] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991.

[23] L. C. Guillou and J.-J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In S. Goldwasser, editor, *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231. Springer-Verlag, 1990, 21–25 Aug. 1988.

[24] S. Halevi and S. Micali. Practical and provably-secure commitment schemes from collision-free hashing. In N. Koblitz, editor, *Advances in Cryptology—CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 201–215. Springer-Verlag, 18–22 Aug. 1996.

[25] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Computing*, 28(4), 1999.

[26] H. Krawczyk and T. Rabin. Chameleon signatures. In *Network and Distributed System Security Symposium*, pages 143–154. The Internet Society, 2000.

[27] Silvio Micali, Michael Rabin, and Joe Kilian. Zero-knowledge sets. In *Proc. 44th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2003.

[28] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):51–158, 1991.

[29] H. Ong and C. P. Schnorr. Fast signature generation with a Fiat Shamir-like scheme. In I. B. Damgård, editor, *Advances in Cryptology—EUROCRYPT 90*, volume 473 of *Lecture Notes in Computer Science*, pages 432–440. Springer-Verlag, 1991, 21–24 May 1990.

[30] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology—EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*. Springer-Verlag, 2–6 May 1999.

[31] A. Shamir and Y. Tauman. Improved online/offline signature schemes. In J. Kilian, editor, *Advances in Cryptology—CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 355–367. Springer-Verlag, 19–23 Aug. 2001.

[32] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.