Collision Resistant Hashing for Paranoids: Dealing with Multiple Collisions

Ilan Komargodski[1(✉)], Moni Naor[2], and Eylon Yogev[2]

[1] Cornell Tech, NewYork, NY 10044, USA
komargodski@cornell.edu
[2] Weizmann Institute of Science, 76100 Rehovot, Israel
{moni.naor,eylon.yogev}@weizmann.ac.il

Abstract. A collision resistant hash (CRH) function is one that compresses its input, yet it is hard to find a collision, i.e. a $x_1 \neq x_2$ s.t. $h(x_1) = h(x_2)$. Collision resistant hash functions are one of the more useful cryptographic primitives both in theory and in practice and two prominent applications are in signature schemes and succinct zero-knowledge arguments.

In this work we consider a relaxation of the above requirement that we call Multi-CRH: a function where it is hard to find $x_1, x_2, \ldots, x_k$ which are all distinct, yet $h(x_1) = h(x_2) = \cdots = h(x_k)$. We show that for some of the major applications of CRH functions it is possible to replace them by the weaker notion of a Multi-CRH, albeit at the price of adding interaction: we show a constant-round statistically-hiding commitment scheme with succinct interaction (committing to $\mathsf{poly}(n)$ bits requires exchanging $\tilde{O}(n)$ bits) that can be opened locally (without revealing the full string). This in turn can be used to provide succinct arguments for any NP statement.

We formulate four possible worlds of hashing-related assumptions (in the spirit of Impagliazzo's worlds). They are (1) *Nocrypt*, where no one-way functions exist, (2) *Unihash*, where one-way functions exist, and hence also UOWHFs and signature schemes, but no Multi-CRH functions exist, (3) *Minihash*, where Multi-CRH functions exist but no CRH functions exist, and (4) *Hashomania*, where CRH functions exist. We show that these four worlds are distinct in a black-box model: we show a separation of CRH from Multi-CRH and a separation of Multi-CRH from one-way functions.

I. Komargodski—Supported in part by a Packard Foundation Fellowship and AFOSR grant FA9550-15-1-0262. Most work done while the author was a Ph.D. student at the Weizmann Institute of Science, supported in part by a grant from the Israel Science Foundation (no. 950/16) and by a Levzion Fellowship.

M. Naor and E. Yogev—Supported in part by a grant from the Israel Science Foundation (no. 950/16). Moni Naor is the incumbent of the Judith Kleeman Professorial Chair.

J. B. Nielsen and V. Rijmen (Eds.): EUROCRYPT 2018, LNCS 10821, pp. 162–194, 2018.
https://doi.org/10.1007/978-3-319-78375-8_6

1 Introduction

In any function that compresses its input, say from $2n$ bits to n bits, there are many collisions, that is, pairs of distinct inputs whose image is the same. But what is the complexity of finding such a collision? Families of functions where this collision finding task is hard are known as collision resistant hash (CRH) functions.[1] CRH functions have many appealing properties, such as preservation under composition and concatenation. The presumed hardness of finding collisions in such functions is the basis for increased efficiency of many useful cryptographic schemes, in particular signature schemes and succinct (zero-knowledge) arguments, i.e., methods for demonstrating the correctness of a statement that are much shorter than the proof or even the statement itself (see Kilian [37] and Barak and Goldreich [4]). The latter is achieved via a hash tree commitment[2] scheme whose opening is local i.e., opening a bit does not require revealing the full string (known as a "Merkle tree"). Results of this sort enable the construction of efficient delegation of computation where the goal is to offload significant computation to some server but also to verify the computation.

Such a task ("breaking a collision resistant hash function") is indeed hard based on a variety of assumptions such as the hardness of factoring integers, finding discrete logs in finite groups or learning with errors (LWE). There are popular functions (standards) with presumed hardness of collision finding such as SHA-2 and SHA-3 (adopted by NIST[3] in 2015). These functions can be evaluated very quickly; however, their hardness is based on more ad hoc assumptions and some former standards have been shown to be insecure (such as MD4, MD5, SHA-1). On the other hand there is no known construction of CRHs based solely on the existence of one-way functions or even one-way permutations and, furthermore, they were shown to be separated in a black-box model (see Simon [50]).

But a sufficiently compressing function also assures us that there are **multiple collisions**, i.e., k distinct values whose image under the function is equal. What about the problem of finding a k-collision? Assuming such hardness is a *weaker* computational assumption than hardness of finding a single pair of colliding inputs and the question is whether it yields a useful primitive.

In this paper we deal with *multiple collision resistant hash* (MCRH) functions and systematically investigate their properties and applications. We show that for some of the major applications of CRH functions it is possible to replace them by an MCRH, albeit at the price of adding some rounds of interaction:

[1] The function $h \in H$ can be sampled efficiently, it is easy to compute $h(x)$ given h and x, however, given h it is hard to find $x_1 \neq x_2$ s.t. $h(x_1) = h(x_2)$.

[2] A commitment scheme is a protocol where a sender commits to a string x in the "commit" phase and that later can be revealed at the opening phase. The two properties are binding and hiding: in what sense is the sender bound to the string x (computationally or information theoretically) and in what sense is x hidden from the receiver before the opening - statistically or computationally.

[3] NIST is the National Institute of Standards and Technology, a US agency.

a constant-round[4] **commitment scheme with succinct communication that can be opened locally** (without revealing the full string) (see Theorem 2). This implies that it is possible to effectively verify the correctness of computation much more efficiently than repeating it. As an application we get universal arguments [4] (and public-coin zero-knowledge argument systems for NP [3]) with an arbitrary super-constant number of rounds based on MCRH functions. We also provide a constant-round statistically-hiding scheme and thus we can get constant-round statistical zero-knowledge arguments [8].[5]

On the other hand, we show various **black-box separation** results concerning MCRH. First, we separate them from one-way permutations. This follows from the lower bound of Haitner et al. [23] on the number of rounds needed to build a statistically-hiding commitment from one-way permutations. Furthermore, we show a black-box separation from standard CRH: there is no fully black-box construction of a k-MCRH from a $(k+1)$-MCRH for all k with polynomial security loss (see Theorem 4). These results yield an infinite hierarchy of natural cryptographic primitives, each two being separated by a fully black-box construction, between one-way function/permutations and collision-resistant hash function.[6]

One motivation for investigating MCRH functions is the progress in finding collisions in the hash function SHA-1 (that has long been considered insecure [53]) and recently an actual meaningful collision has been found [51]. In general, finding a collision with arbitrary Initialization Vector (IV) allows finding multiple collisions in an iteration of the function (say in a Merkle-Damgård lopsided tree), as shown by Joux [35] (see also Coppersmith and Girault et al. [9,15] for older attacks). However, for the compression function of SHA-1 (or other such functions) there is no non-trivial algorithm for finding multi-collisions.[7] Also, multi-collision resistance is sometimes useful for optimizing concrete parameters, as shown by Girault and Stern [16] for reducing the communication in identification schemes. So the question is what can we do if all we assume about a given hash function is that multi-collision are hard to find, rather than plain collisions.

Our interest in MCRH functions originated in the work of Komargodski et al. [39], where MCRHs were first defined in the context of the bipartite Ramsey problem. They showed that finding cliques or independent sets in succinctly-represented bipartite graphs (whose existence is assured by Ramsey Theory) is equivalent to breaking an MCRH: a hard distribution for finding these

[4] By "constant-round" we mean that for a $c \in \mathbb{N}$ to commit to a string of length n^c using a compression function from $2n$ to n bits the number of rounds is a constant that depends on c.

[5] For constant values of k, we give a 4-round computationally-binding commitment scheme with succinct communication (see Theorem 3).

[6] This hierarchy translates to an infinite hierarchy of natural subclasses in TFNP. See the full version [38] for details.

[7] Beyond the "birthday-like" algorithm that will take time $2^{n \cdot \frac{k-1}{k}}$ [35], where 2^n is the size of the range. See [30] for very recent work in the quantum case.

objects implies the existence of an MCRH and vice-versa (with slightly different parameters).[8]

Families of CRHs compose very nicely, and hence *domain extension* is relatively simple, that is, once we have a CRH that compresses by a single bit we can get any polynomial compression (i.e., from $\mathsf{poly}(n)$ to n bits). In contrast, we do not know how to construct a k'-MCRH on very large domains from a fixed k-MCRH, where k' is not much larger than k. Nevertheless, in Sect. 6, we show how to get such a construction with $k' = k^{O(\log n)}$.

A well-known relaxation of CRHs are Universal One-way Hash Functions (UOWHF) or second pre-image resistance: first the target x is chosen (perhaps adversarially), then a function $h \in_R H$ is sampled and the challenge is to find $x' \neq x$ s.t. $h(X) = h(x')$. Such families are good enough for signatures (at least existentially) and can be used for the hash-and-sign paradigm, if one chooses h per message (see Naor and Yung [46] and Mironov [43]). It is known how to get such family of functions from one-way functions, but the construction is rather involved and inefficient (and there are some inherent reasons for that, see [14]). We show how to go from *any* interactive commitment protocol where the communication is shorter than the string committed to (a succinct commitment) to a UOWHF (see Theorem 5). Together with our commitment schemes, this gives new constructions of UOWHFs on long inputs based on MCRHs with a shorter description than the ones known starting from a UOWHF on fixed input length.

The Four Worlds of Hashing. Impagliazzo's five worlds [32] are a way to characterize the strength of a cryptographic assumption. The worlds he defined are: *Algorithmica* (where $\mathsf{P} = \mathsf{NP}$), *Heuristica* (where NP is hard in the worst case but easy on average, i.e., one simply does not encounter hard problems in NP), *Pessiland* (where hard-on-the-average problems in NP exist, but one-way functions do not exist), *Minicrypt* (where one-way functions exist), and *Cryptomania* (where Oblivious Transfer exists). (Nowadays, it is possible to add a sixth world, *Obfustopia*, where indistinguishability obfuscation for all programs exists.)

In the spirit of Impagliazzo's five worlds of cryptographic assumptions, we define four worlds of hashing-related primitives:

Nocrypt: A world where there are no one-way functions. There are no cryptographic commitments of any kind in this world [34].

Unihash: A world where one-way functions exist (and so do UOWHFs), but there are no MCRH functions. Therefore, signatures exist and hashing applications such as the hash-and-sign paradigm [46]. Also, statistically-hiding commitments exist (albeit with a linear number of rounds) [23,24,26,45]. There are no known short commitment (where the communication is much shorter than the string committed to).

[8] In the bipartite Ramsey problem, the goal is to find a bi-clique or bi-independent set of size $n/4 \times n/4$ in a bipartite graph of size $2^n \times 2^n$. The work of [39] showed that if this problem is hard, then there exists an $(n/4)$-MCRH from n bits to $n/2$ bits. Conversely, If a $\sqrt{n}$-MCRH mapping n bits to $\sqrt{n}/8$ bits exists, then this problem is hard.

Minihash: A world where MCRH exists but there is no CRH: that is, for some polynomial $k(n)$ there exists a k-MCRH that compresses $2n$ to n bits. In this work we give a protocol for short and statistically-hiding commitments with a *constant* number of rounds. Furthermore the string can be opened locally, with little communication and computation, without requiring the full opening of the string.

Hashomania: A world where CRH exists. There is a short commitment protocol that requires only *two* rounds (i.e., two messages) with local opening. This is the famed Merkle-tree.

Note that our separation results imply that these four worlds have black-box separations. Unihash and Minihash are separated by the separation of MCRH from one-way permutations, and Minihash and Hashomania are separated by the separation of CRH from MCRH. Moreover, the separation in Sect. 7 actually implies that the world *Minihash* can be split further into sub-worlds parameterized by k, the number of collisions it is hard to find.

Multi-pair Collision Resistance. A different way to relax the standard notion of collision resistance is what we call *multi-pair-collision-resistance*, where the challenge is to find *arbitrary* k distinct pairs of inputs that collide (possibly to different values). One may wonder what is the difference between these two notions and why we focus on the hardness of finding a k-wise collision rather than hardness of finding k distinct colliding pairs. The answer is that the notion of k-pair-collision-resistance is *existentially equivalent* to the standard notion of collision resistance (see the full version [38] for details).

Concurrent work

In parallel to this work, MCRH functions were studied by two other groups that obtained various related results [6,7] with different motivation and perspective.

Berman et al. [6] showed how to obtain MCRH functions from a specific assumption. Concretely, they construct an n^2-MCRH function compressing inputs of length n to outputs of length $n - \sqrt{n}$ from the average-case hardness of the min-max variant of the entropy approximation problem. This variant is a promise problem where the YES inputs are circuits whose output distribution has *min*-entropy at least κ, whereas NO inputs are circuits whose output distribution has *max*-entropy less than κ.[9] Berman et al. also show how to get a constant-round statistically-hiding (computationally-binding) commitment scheme from any k-MCRH that compresses n bits into $n - \log k$ bits (which implies a black-box separation of MCRH from one-way permutations). However, their commitment scheme is not short and does not support local opening[10].

[9] The original *entropy approximation* problem is the one obtained by replacing the min- and max-entropy with the Shannon entropy. It is known to be complete for the class of languages that have non-interactive statistical zero-knowledge proofs (NISZK) [17].

[10] Starting out with such a weak primitive our methods will not yield a succinct commitment either.

Bitansky et al. [7] replace the CRH assumption with a k-MCRH in several applications related to zero-knowledge and arguments of knowledge for NP with few rounds. They rely on either MCRH that compresses by a polynomial factor (say n^2 bits into n), or alternatively on an MCRH that compresses by a linear factor but lose a quasi-polynomial factor in security. Their main technical component is a two-round (i.e., two-message) short commitments with local opening but with *weak* computational-binding. The latter means that the sender may be able to open the commitment to more than one value, but not to too many values. Their construction of the commitment scheme is related to our construction presented in Theorem 3 but they design a specific code that allows them to get local opening.

Summary of Results and Paper Organization
Our main results are:

1. Any k-MCRH can be used to get a constant round short commitment scheme which is computationally-binding, statistically-hiding and support local opening (à la Merkle commitments). This result in Sect. 5.
2. Any k-MCRH, where k is constant, can be used to get a 4 round short commitment scheme which is computationally-binding and statistically-hiding opening. This appears in Sect. 6.
3. We prove a fully black-box separation between standard collision resistant hash functions and multi-collision resistant ones. This appears in Sect. 7.
4. We present a generic and direct construction of UOWHFs from any short commitment schemes (and thereby from any multi-collision resistant functions). See Sect. 8.

In Sect. 2 we provide an overview of our main ideas and techniques. In Sects. 3 and 4 we provide preliminary standard definitions used throughout the paper and the definition of MCRH functions, respectively.

2 Our Techniques

In this section we present some of our main ideas and techniques used in the construction of the commitment scheme and in the black-box separation result.

2.1 The Main Commitment Scheme

A commitment scheme is a two stage interactive protocol between a sender and a receiver such that after the first stage the sender is bound to at most one value (this is called "binding"). In the second stage the sender can open his committed value to the sender. There are a few security properties one can ask from such a protocol: have statistical/computational binding, and have the committed value of the sender be statistically/computationally hidden *given* the commitment (this is called "hiding"). Our commitment scheme satisfies computational binding and statistical hiding. In this overview we will mostly focus on obtaining computational binding and briefly discuss how we obtain hiding towards the end.

There is a trivial commitment protocol that is (perfectly) binding: let the sender send its value to the receiver. Perhaps the most natural non-trivial property one can ask is that the commitment is *shorter* than the committed string. There are additional useful properties one can require such as *local-opening* which allows the sender to open a small fraction of its input without sending the whole string (local opening is very important in applications of such commitment schemes, e.g., to proof systems and delegation protocols). In our protocol the commitment is short and it supports local-opening; we will focus here on the former and shortly discuss the latter towards the end.

Our goal now is to construct a commitment scheme which is computationally binding and the commitment is shorter than the value of the sender. If we had a standard collision resistant hash function mapping strings of length $2n$ to strings of length n, this task would be easy to achieve: the receiver will sample a hash function h and send it to the sender which will reply with $h(x^*)$, where $x^* \in \{0,1\}^{2n}$ is its value. The commitment thus consists of $(h, h(x^*))$ and its size is n bits.[11] It is easy to verify that for a sender to cheat during the opening phase it actually has to break the collision resistance of h (i.e., come up with a value $x \neq x^*$ such that $h(x) = h(x^*)$).

When h is only a k-MCRH for $k > 2$, the above protocol is clearly insecure: the sender can potentially find two inputs that collide to the same value and cheat when asked to open its commitment. The first observation we make is that even though the sender is not bound to a single value after sending $h(x^*)$, it is bound to a set of values of size at most $k-1$. Otherwise, at least intuitively, he might be able to find k inputs that map to the same output relative to h, which contradicts the security of the k-MCRH. Our first idea is to take advantage of this fact by adding an additional round of communication whose goal is to "eliminate" all but one possible value for the sender. Specifically, after the receiver got $h(x^*)$, it samples a random universal hash[12] function, g, mapping strings of length $2n$ to strings of length m. and sends it to the sender. The sender then responds with $g(x)$. The commitment thus consists of $(h, h(x^*), g, g(x^*))$. To open the commitment the sender just sends x^* (just as before).

One can show that this protocol is binding: Out of the $k-1$ possible values the sender knows that are consistent with $h(x^*)$, with probability roughly $k^2 \cdot 2^{-m}$ there will be no two that agree on $g(x^*)$. Conditioning on this happening, the sender cannot submit $x \neq x^*$ that is consistent with both $h(x^*)$ and $g(x^*)$. To formally show that the protocol is binding we need to show how to find a k-wise collision using a malicious sender. We simulate the protocol between the malicious sender and the receiver *multiple times* (roughly k times) with the same h but with freshly sampled g, by *partially rewinding* the malicious sender. We show that with good probability, every iteration will result with a new collision.

[11] For simplicity of presentation here, we ignore the cost of the description of h, so it will not significantly affect the size of the commitment.

[12] A universal hash function is a function of families $\mathcal{G} = \{g\colon \{0,1\}^n \to \{0,1\}^m\}$ such that for any $x, y \in \{0,1\}^n$ such that $x \neq y$ it holds that $\Pr_{g \leftarrow \mathcal{G}}[g(x) = g(y)] \leq 2^{-m}$.

The protocol consists now of 4 rounds, but is the commitment short? Well, it depends on m and on the description size of g. The description size of a universal function is proportional to the input size ($2n$ in our case), which totally ruins the shortness of the protocol. We fix this by sampling g from an *almost* universal family and apply the above protocol. We obtain a 4-round protocol in which the commitment size is of the order roughly $n + m + \log(1/\delta)$, where δ is related to the error probability of the almost universal function. Choosing m and δ appropriately we obtain a protocol with short ($<2n$) commitments.

Handling Longer Inputs. How would we commit on a longer string, say of $10n$ bits or even n^{10} bits? (Recall that all we have is a hash function mapping $2n$ bits into n bits.) One well-known solution is based on a standard collision resistant hash function, and what is known as a Merkle tree (the tree structure will be useful later for a local opening). The input $x \in \{0,1\}^{2^d n}$ (for simplicity think of d as either a large constant or even $O(\log n)$) is partitioned into 2^d blocks each of size n. These blocks are partitioned into pairs and the hash function is applied to each pair resulting in 2^{d-1} blocks. Then, the remaining blocks are partitioned into pairs and the hash function is applied on each pair. This is repeated d times resulting in a binary tree of hash values of depth d. The value associated with the root of the tree is called the root-hash.

If h is a standard CRH sent by the receiver, then it is known that sending the root-hash by the sender is actually a commitment on the input x^* [37,41]. Can we apply the same trick from before to make this a commitment protocol even when h is only a k-MCRH? That is, after sending the root-hash, let the sender sample a good-enough combinatorial hash function $g\colon \{0,1\}^{2^d n} \to \{0,1\}^m$ and send it to the sender that will reply with $g(x^*)$. Is this protocol binding? The answer is "no", even for large values of m. Observe that for every node in the Merkle tree, the sender can potentially provide $k-1$ valid inputs (that hash to the same value). Since the tree is of depth d, one can observe that by a mix-and-match method of different colliding values on different nodes of the tree the sender might be able to come up with as many as $(k-1)^{2^d}$ valid inputs x whose corresponding root-hash is $h(x^*)$. Thus, to satisfy that no two have the same value under g, we have to choose $m \approx 2^d \cdot \log(k-1)$, in which case the almost uniform hash function has a pretty long description. Nevertheless, it is less than $2^d n$ (the input length) so we might hope that some progress has been made. Is this protocol computationally-binding? Not quite. Using the proof technique from above (of partially rewinding and "collecting" collisions) would require running the malicious sender more than $(k-1)^{2^d}$ times until it has to present more than $k-1$ collisions for some value. This is, of course, way too expensive.

The bottom line of the above paragraph is that "mix-and-match" attacks are very powerful for a malicious sender in the context of tree hashing by allowing him to leverage the ability of finding few collisions into an ability to find exponentially many collisions. The reason why this happens is that we compose hash functions but apply the universal hash function on the whole input as a single

string. Our next idea is to apply a "small" hash function $g\colon \{0,1\}^{2n} \to \{0,1\}^n$ per node in the Merkle tree. That is, after the sender sends the root-hash $h(x^*)$, the receiver samples g and sends it to the sender. The sender computes $g(\cdot,\cdot)$ for every pair of siblings along the Merkle tree, concatenates them all and sends this long string back to the receiver. This protocol is more promising since, in some sense, we have a small consistency check per node in the tree which should rule out simple "mix-and-match" attacks. This is our construction and the proof of security works by partially rewinding a malicious sender and "collecting" collisions until we get k collisions with respect to some internal node in the tree (we need to collect roughly $2^d k$ collisions overall so that such a node exists, by the pigeonhole principle). How efficient is the protocol? Details follow.

The protocol still consists of 4 rounds. A commitment consists of the hash function h, the root hash $h(x^*)$, a universal hash function $g\colon \{0,1\}^{2n} \to \{0,1\}^m$ and the value of g on *every* internal node of the tree. The overall size is thus of order $n + 2^d m$. Notice that $n + 2^d m \ll 2^d n$ whenever d is not too small, so we have made progress! We reduced the size of the commitment by a factor of m/n. The final step is to really get down to a commitment of size roughly n. To achieve this, we apply our protocol *recursively*: Instead of sending the hashes (with respect to g) of all internal nodes, we run our *commit* protocol recursively on this string. Notice that this string is shorter ($2^d m$ compared to $2^d n$) so the recursion does progress. The base of the recursion is when the string length is roughly n bits, then the sender can simply send it to the receiver.

Choosing the parameters carefully, we get various trade-offs between the number of rounds, the commitment size, and the security of the resulting protocol. For example, setting $m = n^{0.99}$, results with a $O(1)$-round protocol in which the commitment size is $O(n)$ (here the big "O" hides constants that depend on $\log_n(|x^*|)$ which is constant for a polynomially long x^*), and whose security is worse than the security of the k-MCRH by an additive factor of $\exp(-n^{0.99})$.

Local Opening. Due to the tree structure of our commitment protocol, it can be slightly modified to support local opening. Recall that the goal here is to allow the receiver to send an index i of a block to the sender, who can reply with the opening of the block, with communication proportional to n but not to the number of blocks 2^d. The idea here is, given an index i of a block, to open the hash values along the path corresponding to the i-block along with the tree sibling of every node in the path. Then, i' is defined to be the index of the block in the shorter string (the string committed to in the next step of the recursion) which containing all the $g(\cdot,\cdot)$ values of the nodes on the path (we make sure that such a block exists). Then, we add the hash values of the path for block i' and continue in a recursive manner.

Statistical Hiding. We show how to transform any short commitment scheme that is computationally binding (but perhaps not hiding) to a new scheme that is short, computationally binding and *statistically hiding*. Moreover, if the original scheme admits a local-opening, then the new scheme admits a local-opening

as well. Our transformation is information theoretic, adds no additional assumptions and preserves the security, the number of rounds and communication complexity of the original scheme (up to a small constant factor). The transformation is partially based on ideas originating in the work of Naor and Yung [46, Sect. 5.2] and the follow-up works of Damgård, Pedersen, and Pfitzmann [11,12] giving constructions of statistical-hiding commitments from (standard) collision resistant hash function.

The idea of our transformation is to leverage the fact that the basic commitment protocol is short: when committing to a long string x^*, the communication is very short. Thus, a large portion of x^* is not revealed to the receiver by the protocol so this part of x^* is statistically hidden. The task that remains is to make sure that all of x^* is hidden. Thus, instead of committing to x^* directly, we commit to a random string r that is independent of x^* and slightly longer. Then, we extract from r the remaining randomness r' *given the communication of the protocol* using a strong extractor. Finally, we commit on the string $x^* \oplus r'$. It is not hard to show that if the original scheme was computationally binding, then the new one is as well. The fact that the scheme is statistically-hiding follows from the use of the strong extractor and the fact that the commitment is short.

One problem with the recipe above, is that the protocol (as describe) no longer admits a local-opening. This is because to open an index i, we need the i-th output bit of the extractor, but computing this bit might require reading a large portion of the input of r. Our solution is to break the input to sufficiently small parts such that each part is small enough to fit in a local-opening but is long enough to have enough entropy (given the communication of the protocol) so that we can apply the extractor on it.

2.2 Separating Multi-CRH from Standard CRH

We show barriers of constructing a collision-resistant hash function from a 3-multi-collision-resistant hash function. We rule out fully black-box constructions (see Definition 9). Our proof technique is inspired by the works of Asharov and Segev [2] and Haitner et al. [23], that are based in turn on ideas originating in the works of Simon [50], Gennaro et al. [14] and Wee [54]. However, when trying to adapt their proof to the setting of multi collisions one encounters several obstacles and we explain how to overcome them.

The high-level overview of the proof is to show that there exists an oracle Γ such that relative to Γ there exists a 3-MCRH, however there exist no standard CRH. Our oracle will contain a truly random function f that maps $2n$ bits to n bits. Relative to this oracle, it is clear that 3-MCRH exists, however, also standard CRH exist. We add an oracle $\mathsf{ColFinder}$ that will be used to break any CRH construction. The main difficulty of the proof is to show that this oracle cannot be used to break the 3-MCRH.

The oracle $\mathsf{ColFinder}$ is essentially the same as in Simon [50]. It gets as an input a circuit C, possibly with f gates and it outputs two random elements w, w' such that $C(w) = C(w')$. It is easy to see that no family of hash functions can be collision resistant in the presence of such an oracle. A single call to $\mathsf{ColFinder}$

with the query C (where $C(x) = f(x)$) will find a collision with high probability. The main question is whether this oracle be used to find *multiple* collisions?

Originally, Simon showed that this oracle cannot be used to *invert* a one-way function (or even a permutation). Let $\mathcal{A}$ be an adversary that uses ColFinder to invert f on a random challenge $y = f(x)$. Clearly, if $\mathcal{A}$ make no calls to ColFinder then his chances in inverting y are negligible. Assume, for simplicity, that $\mathcal{A}$ performs only a single query to ColFinder. In order for $\mathcal{A}$ to gain some advantage, it must make an "interesting" query to ColFinder. That is, a query which results in w, w' and the computation of either $C(w)$ or $C(w')$ makes a direct query to some $x \in f^{-1}(y)$. This event is called a hit. An important point is that for any circuit C the marginal distribution of w and of w' is uniform. Therefore, the probability of the event "hit" in the ColFinder query is at most twice that probability when evaluating $C(z)$ for a random z. Thus, we can construct a simulator that replaces $\mathcal{A}$'s query to ColFinder with the evaluation of $C(z)$ and hits an inverse of y with roughly the same probability as $\mathcal{A}$ (while making no queries to ColFinder). The task of inverting y without ColFinder can be shown to be hard, ruling out the existence of such a simulator and in turn of such an adversary $\mathcal{A}$.

Our goal is to extend this proof and show that ColFinder cannot be used to find 3-wise collisions. The above approach above simply does not work: specifically, in our case the event "hit" corresponds to query C to ColFinder that results in w, w' and the computation of $C(w)$ and $C(w')$ *together* make direct queries three elements x_1, x_2, x_3 that collide under f (i.e., $f(x_1) = f(x_2) = f(x_3)$). It might be the case that these three elements are hit by $C(w)$ and $C(w')$ combined, but never by one of them alone. Thus, when simulating the $C(z)$ for a random z we will hit only part of the trio x_1, x_2, x_3 and might never hit all three.

Our main observation is that since $\mathcal{A}$ finds a 3-wise collision, but ColFinder finds only a 2-wise collision (namely w, w'), by the pigeonhole principle, either w or w' will hit two of the three elements of the 3-wise collision. Again, since the marginals of w and w' each are uniform, we can construct a simulator that runs $\mathcal{A}$ and on the query to ColFinder samples a uniform z and compute $C(z)$ and will get a colliding x_1 and x_2 without performing any queries to ColFinder. Then, one can show that such a simulator cannot exist.

Several problems arise with this approach. First, notice that this does not extend to an adversary $\mathcal{A}$ that makes more than one query. In such a case, the resulting simulator finds a 2-wise collision x_1, x_2 without the "hit" event occurring (i.e., finding a 3-wise collision) but while performing several ColFinder queries. Such a simulator (that finds a collision), of course, *trivially* exists, and we do not get the desired contradiction. Nevertheless, we show that the collision found by our simulator is somewhat special, and using ColFinder one can only find "non-special" collisions, ruling out the existence of $\mathcal{A}$ in this case. Second, the event "hit" itself might be spread out within several ColFinder queries, where, for example, one queries finds x_1 and then another query finds x_2, x_3. We tailor a simulator for each case, and show that for each case the resulting simulating cannot exist, completely ruling out the possibility of $\mathcal{A}$ to exist.

3 Preliminaries

Unless stated otherwise, the logarithms in this paper are base 2. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. We denote by U_n the uniform distribution over n-bit strings. For a distribution $\mathcal{D}$ we denote by $x \leftarrow \mathcal{D}$ an element chosen from $\mathcal{D}$ uniformly at random. We denote by $\circ$ the string concatenation operation. A function $\mathsf{negl}\colon \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for every constant $c > 0$, there exists an integer N_c such that $\mathsf{negl}(n) < n^{-c}$ for all $n > N_c$.

Definition 1 (Statistical Distance). *The statistical distance between two random variables X, Y is defined by*

$$\Delta(X, Y) \triangleq \frac{1}{2} \cdot \sum_x |\Pr[X = x] - \Pr[Y = x]|$$

We say that X and Y are δ-close (resp. -far) if $\Delta(X, Y) \leq \delta$ (resp. $\Delta(X, Y) \geq \delta$).

3.1 Limited Independence

Definition 2 (k-wise independence). *Fix $n, m, k \in \mathbb{N}$. A function family $\mathcal{G} = \{g\colon \{0,1\}^n \to \{0,1\}^m\}$ is k-wise independent if for every distinct $x_1, \ldots, x_k \in \{0,1\}^n$ and every $y_1, \ldots, y_k \in \{0,1\}^m$ it holds that*

$$\Pr_{g \leftarrow \mathcal{G}}[\forall i \in [k]\colon g(x_i) = y_i] = \frac{1}{2^{km}}.$$

It is known that for every $m \leq n$, there exists a k-wise independent family of functions, where each function is described by $k \cdot n$ bits. One well-known construction which is optimal in terms of size is by letting each $g \in \{0,1\}^{k \cdot n}$ describe a degree $k-1$ polynomial over $\mathsf{GF}[2^n]$. The description of the polynomial requires k field elements so $k \cdot n$ bits are enough. Evaluation of such a function is merely an evaluation of the polynomial.

In some applications (including some of ours) the input size n is very large and we prefer that the description size of the hash function to be much shorter. To circumvent this, it is sometimes enough to use *almost* k-wise independent functions.

Definition 3 (Almost k-wise independence). *Fix $n, m, k \in \mathbb{N}$ and $\delta \in \mathbb{R}$. A function family $\mathcal{G} = \{g\colon \{0,1\}^n \to \{0,1\}^m\}$ is (k, δ)-wise independent if for every distinct $x_1, \ldots, x_k \in \{0,1\}^n$ the distribution of $(g(x_1), \ldots, g(x_k))$ is δ-close to the distribution $(u_1, \ldots, u_k)$, where $g \leftarrow \mathcal{G}$ and each $u_i \leftarrow \{0,1\}^m$ are chosen uniformly at random.*

It is known that for every $m \leq n$, there exists a (k, δ)-wise independent function with each function $g \in \mathcal{G}$ being described by $O(mk + \log(n/\delta))$ bits [1,44] (see also [52]).

3.2 Randomness Extractors

We consider random variables supported on n-bit strings. A random variable X is said to have min-entropy $\mathsf{H}_\infty(X) = k$ if for every $x \in \mathsf{Supp}(X)$ it holds that $\Pr[X = x] \leq 2^{-k}$.

We say that a function $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ϵ)-*seeded extractor* if for every distribution X over $\{0,1\}^n$ with min-entropy k, it holds that

$$\Delta(\mathsf{Ext}(X, U_d), U_m) \leq \epsilon.$$

The extractor Ext is said to be *strong* if $\mathsf{Ext}'(x,s) = \mathsf{Ext}(x,s) \circ s$ is a (k,ϵ)-seeded extractor. That is, if

$$\Delta((\mathsf{Ext}(X, U_d) \circ U_d), (U_m \circ U_d)) \leq \epsilon.$$

The famous leftover hash lemma [27,33] says that a pairwise independent function family is a strong extractor.

Proposition 1. *Let $\mathcal{G} = \{g\colon \{0,1\}^n \to \{0,1\}^m\}$ be a pairwise independent family of hash functions where $m = k - 2\log(1/\epsilon)$. Then, $\mathsf{Ext}(x,h) = h(x)$ is a strong (k,ϵ)-seeded extractor.*

Note that the seed length in this extractor equals the number of bits required to sample $g \leftarrow \mathcal{G}$ which is $2n$ bits.

We will also need the following standard proposition that says that conditioning does not reduce entropy by more than the information given by the condition.

Proposition 2. *Let X and Y be random variables. Then, if Y is supported on strings of length k, then $\mathsf{H}_\infty(X \mid Y) \geq \mathsf{H}_\infty(X) - k$.*

3.3 List-Recoverable Codes

The classical notion of error correcting codes ensures that for a code $C \subseteq \mathbb{F}^n$, where $\mathbb{F}$ is a finite field, given a somewhat corrupted version of $c \in C$, it is possible to recover c. The model of allowed corruptions is that some fraction of the symbols in the codeword might be adversarially changed. List recoverable codes were introduced to handle a different model of corruptions: they allow an adversary to submit, for every coordinate $i \in [n]$ a small list $S_i \subseteq \mathbb{F}$ of possible symbols. In this model, it is impossible to completely recover a codeword given the lists, but these codes guarantee that there is only a small list of codewords that are consistent with all the lists.

More precisely, a mapping $C\colon \mathbb{F}^k \to \mathbb{F}^n$ from length k messages to length n codewords, is called (α, ℓ, L)-*list-recoverable* if there is a procedure that is given a sequence of lists $S_1, \ldots, S_n \subseteq \mathbb{F}$ each of size ℓ, and is able to output all messages $x \in \mathbb{F}^k$ such that $C(x)_i \notin S_i$ for at most an α fraction of the coordinates $i \in [n]$. The code guarantees that there are at most L such messages.

Definition 4 (List-recoverable codes). *Let $\alpha \in [0,1]$. We say that a tuple $x \in (\{0,1\}^k)^n$ is* α-consistent *with sets $S_1, \ldots, S_n \subseteq \{0,1\}^k$, if $|\{i : x_i \in S_i|\} \geq \alpha n$.*

A function $C\colon \{0,1\}^v \to (\{0,1\}^k)^n$ is (α, ℓ, L)-list recoverable, if for every set $S_1, \ldots, S_n \subseteq \{0,1\}^k$ each of size at most ℓ, there are at most L strings $x \in \{0,1\}^v$ such that $C(x)$ is α-consistent with $S_1, \ldots, S_n$. For $\alpha = 1$, we omit α in the above notation and call C (ℓ, L)-list recoverable. The strings in the image of C are referred to as codewords.

These code were initially studied in the context of *list-decoding* (and indeed the latter is just a special case of the former with $\ell = 1$) by [18–21]. More recently, they were proven useful in other areas such as compressed sensing [47], non-adaptive domain extension for hashing [25], domain extension for public random functions and MACs [13,40], and more (see Sect. 6, and for example, [29] and references therein).

A natural relaxation of the above codes is to require that $S_1 = \ldots = S_n$. This variant is called *weakly* list-recoverable codes. A list-recoverable code is immediately weakly list-recoverable and the converse also holds albeit with a minor loss in parameters: An (ℓ, L)-weakly list-recoverable code is an (ℓ, nL)-list-recoverable code. Looking ahead, this loss will not make much of a difference for us since our L will be polynomial in n.

For our purposes, we will need a list-recoverable code with $\alpha = 1$. It is well-known (see e.g., [25]) that the notion of weakly list-recoverable codes is equivalent to unbalanced expanders with a certain expansion property. The left set of vertices in the graph is $\{0,1\}^v$, the right set of vertices is $\{0,1\}^k$ and the left degree is n. This graph naturally induces a mapping $C\colon \{0,1\}^v \to (\{0,1\}^k)^n$ which on input $x \in \{0,1\}^v$ (left vertex) outputs n neighbors (right vertices). The mapping C is (ℓ, L)-list-recoverable iff for every set $S \subseteq \{0,1\}^k$ of size larger than L of nodes on the right, the set of left neighbors of S is of size larger than ℓ.

The following instantiation of locally-recoverable codes based on the explicit construction of unbalanced expanders of [22] is taken (with minor modifications) from [25].

Theorem 1 ([22,25]). *For every $\alpha \geq 1/2$, and $k < v$, there exists a* $\mathsf{poly}(n)$*-time computable function $C\colon \{0,1\}^v \to \left(\{0,1\}^k\right)^n$ for $n = O(v \cdot k)^2$ which defines an (α, ℓ, L)-list recoverable code for every $L \leq 2^{k/2}$ and $\ell = \Omega(L)$. The list-recovery algorithm runs in time* $\mathsf{poly}(v, \ell)$.

3.4 Cryptographic Primitives

A function f, with input length $m_1(n)$ and outputs length $m_2(n)$, specifies for every $n \in \mathbb{N}$ a function $f_n\colon \{0,1\}^{m_1(n)} \to \{0,1\}^{m_2(n)}$. We only consider functions with polynomial input lengths (in n) and occasionally abuse notation and write $f(x)$ rather than $f_n(x)$ for simplicity. The function f is computable in polynomial time (efficiently computable) if there exists an algorithm that for any $x \in \{0,1\}^{m_1(n)}$ outputs $f_n(x)$ and runs in time polynomial in n.

A function family ensemble is an infinite set of function families, whose elements (families) are indexed by the set of integers. Let $\mathcal{F} = \{\mathcal{F}_n : \mathcal{D}_n \to \mathcal{R}_n\}_{n\in\mathbb{N}}$ stand for an ensemble of function families, where each $f \in \mathcal{F}_n$ has domain $\mathcal{D}_n$ and range $\mathcal{R}_n$. An efficient function family ensemble is one that has an efficient sampling and evaluation algorithms.

Definition 5 (Efficient function family ensemble). *A function family ensemble $\mathcal{F} = \{\mathcal{F}_n : \mathcal{D}_n \to \mathcal{R}_n\}_{n\in\mathbb{N}}$ is efficient if:*

- *$\mathcal{F}$ is samplable in polynomial time: there exists a probabilistic polynomial-time machine that given 1^n, outputs (the description of) a uniform element in $\mathcal{F}_n$.*
- *There exists a deterministic algorithm that given $x \in \mathcal{D}_n$ and (a description of) $f \in \mathcal{F}_n$, runs in time $\mathsf{poly}(n, |x|)$ and outputs $f(x)$.*

Universal One-Wayness. A one-way function is an efficiently computable function which is hard to invert on a random output for any probabilistic polynomial-time machine. A universal one-way hash function (UOWHF) is a family of compressing functions $\mathcal{H}$ for which any PPT adversary has a negligible chance of winning in the following game: the adversary submits an x and gets back a uniformly chosen $h \leftarrow \mathcal{H}$. The adversary wins if it finds an $x' \neq x$ such that $h(x) = h(x')$. UOWHF were introduced by Naor and Yung [46] and were shown to imply secure digital signature schemes. Rompel [48] (see also [36]) showed how to construct UOWHF based on the minimal assumption that one-way functions exist.

Definition 6 (Universal one-way hash functions (UOWHF)). *An efficient function family ensemble $\mathcal{F} = \{\mathcal{F}_n : \{0,1\}^{m_1(n)} \to \{0,1\}^{m_2(n)}\}_{n\in\mathbb{N}}$ is a universal one-way hash function family if the probability of every probabilistic polynomial-time adversary $\mathcal{A}$ to win in the following game is negligible in n:*

1. *$\mathcal{A}$, given 1^n, submits $x \in \{0,1\}^{m_1(n)}$.*
2. *Challenger responds with a uniformly random $f \leftarrow \mathcal{F}_n$.*
3. *$\mathcal{A}$ (given f) outputs $x' \in \{0,1\}^{m_1(n)}$.*
4. *$\mathcal{A}$ wins iff $x \neq x'$ and $f(x) = f(x')$.*

3.5 Commitment Schemes

A commitment scheme is a two-stage interactive protocol between a sender $\mathcal{S}$ and a receiver $\mathcal{R}$. The goal of such a scheme is that after the first stage of the protocol, called the commit protocol, the sender is bound to at most one value. In the second stage, called the opening protocol, the sender opens its committed value to the receiver. We also require that the opening protocol allows to open only a single bit of the committed string. More precisely, a commitment scheme for a domain of strings $\{0,1\}^\ell$ is defined via a pair of probabilistic polynomial-time algorithms $(\mathcal{S}, \mathcal{R}, \mathcal{V})$ such that:

- The commit protocol: $\mathcal{S}$ receives as input the security parameter 1^n and a string $s \in \{0,1\}^\ell$. $\mathcal{R}$ receives as input the security parameter 1^n. At the end of this stage, $\mathcal{S}$ outputs $\mathsf{decom}_1 \ldots, \mathsf{decom}_\ell$ (the local decommitments) and $\mathcal{R}$ outputs com (the commitment).
- The local-opening procedure: $\mathcal{V}$ receives as input the security parameter 1^n, a commitment com, an index $i \in [\ell]$, a local-decommitment decom_i, and outputs either a bit b or $\perp$.

A commitment scheme is *public coin* if all messages sent by the receiver are independent random coins.

Denote by $(\mathsf{decom}_1, \ldots, \mathsf{decom}_\ell, \mathsf{com}) \leftarrow \langle \mathcal{S}(1^n, s), \mathcal{R} \rangle$ the experiment in which $\mathcal{S}$ and $\mathcal{R}$ interact with the given inputs and uniformly random coins, and eventually $\mathcal{S}$ outputs a list of ℓ decommitment strings and $\mathcal{R}$ outputs a commitment. The completeness of the protocol says that for all $n \in \mathbb{N}$, every string $s \in \{0,1\}^\ell$, every tuple $(\mathsf{decom}_1, \ldots, \mathsf{decom}_\ell, \mathsf{com})$ in the support of $\langle \mathcal{S}(1^n, s), \mathcal{R} \rangle$, and every $i \in [\ell]$, it holds that $\mathcal{V}(i, \mathsf{decom}_i, \mathsf{com}) = s_i$.

Below we define two security properties one can require from a commitment scheme. The properties we list are *statistical-hiding* and *computational-binding*. These roughly say that after the commit stage, the sender is *bound* to a specific value which remains statistically hidden for the receiver.

Definition 7 (ϵ-binding). *A commitment scheme $(\mathcal{S}, \mathcal{R}, \mathcal{V})$ is $(t(n), \epsilon(n))$-binding if for every probabilistic adversary $\mathcal{S}^*$ that runs in time at most $t(n)$, it holds that*

$$\Pr\left[\begin{array}{c}(i, \mathsf{decom}_i, \mathsf{decom}'_i, \mathsf{com}) \leftarrow \langle \mathcal{S}^*(1^n), \mathcal{R} \rangle \text{ and} \\ \perp \neq \mathcal{V}(i, \mathsf{decom}_i, \mathsf{com}) \neq \mathcal{V}(i, \mathsf{decom}'_i, \mathsf{com}) \neq \perp\end{array}\right] \leq \epsilon(n)$$

for all sufficiently large n, where the probability is taken over the random coins of both $\mathcal{S}^*$ and $\mathcal{R}$.

Given a commitment scheme $(\mathcal{S}, \mathcal{R}, \mathcal{V})$ and an adversary $\mathcal{R}^*$, we denote by $\mathsf{view}_{\langle \mathcal{S}(s), \mathcal{R}^* \rangle}(n)$ the distribution on the view of $\mathcal{R}^*$ when interacting with $\mathcal{S}(1^n, s)$. The view consists of $\mathcal{R}^*$'s random coins and the sequence of messages it received from $\mathcal{S}$. The distribution is take over the random coins of both $\mathcal{S}$ and $\mathcal{R}$. Without loss of generality, whenever $\mathcal{R}^*$ has no computational restrictions, we can assume it is deterministic.

Definition 8 (ρ-hiding). *A commitment scheme $(\mathcal{S}, \mathcal{R}, \mathcal{V})$ is $\rho(n)$-hiding if for every (deterministic) adversary $\mathcal{R}^*$ and every distinct $s_0, s_1 \in \{0,1\}^\ell$, it holds that*

$$\Delta\left(\{\mathsf{view}_{\langle \mathcal{S}(s_0), \mathcal{R}^* \rangle}(n)\}, \{\mathsf{view}_{\langle \mathcal{S}(s_1), \mathcal{R}^* \rangle}(n)\}\right) \leq \rho(n)$$

for all sufficiently large $n \in \mathbb{N}$.

Complexity Measures. The parameters of interest are (1) the number of rounds the commit protocol requires, (2) the size of a commitment, and (3) the size of a local opening.

The *size* of a commitment is the size (in bits) of the output of $\mathcal{S}$ denoted above by com. A *short commitment* is such that the size of com is much smaller than ℓ. Preferably, the size of a short commitment depends solely on n, but poly-logarithmic dependence on ℓ is also okay. The size of a local opening is the maximum size of decom_i (in bits). A protocol is said to support local opening if this size depends only on n and at most poly-logarithmically on ℓ.

3.6 Fully Black-Box Constructions

We give a definition of a fully black-box reduction from an MCRH to standard CRH. For this, we generalize the definition of an MCRH to the setting of oracle-aided computation: The generation and evaluation algorithms of an MCRH are given access to an oracle Γ relative to which they can generate a description of a hash function and evaluate an index at a point. The adversary is also given oracle access to Γ in the security game and has to find multiple collisions relative to it.

We focus here on k-MCRH functions with $k = 3$. The following definition of a "black-box construction" is directly inspired by those of [2,23].

Definition 9. *A fully black-box construction of a collision-resistant function family $\mathcal{H}'$ from a 3-MCRH function family $\mathcal{H}$ mapping $2n$ bits to n bits consists of a pair of probabilistic polynomial-time algorithms $(\mathcal{H}.\mathsf{G}, \mathcal{H}.\mathsf{E})$ and an oracle-aided polynomial-time algorithm M such that:*

- **Completeness:** *For any $n \in \mathbb{N}$, for any 3-MCRH function family $\mathcal{H}$ and any function h produced by $h \leftarrow \mathcal{H}'^{\mathcal{H}}(1^n)$, it holds that $h^{\mathcal{H}} \colon \{0,1\}^{2n} \to \{0,1\}^n$.*
- **Black-box proof of security:** *For any collision resistant hash $\mathcal{H}'$, any probabilistic polynomial-time oracle-aided algorithm $\mathcal{A}$, every polynomial $p(\cdot)$, if*

$$\Pr\left[\begin{matrix} x_1 \neq x_2 \\ h^{\mathcal{H}}(x_1) = h^{\mathcal{H}}(x_2) \end{matrix} \middle| \begin{matrix} h \leftarrow h^{\mathcal{H}}(1^n) \\ (x_1, x_2) \leftarrow \mathcal{A}^{\mathcal{H}}(1^n, h) \end{matrix}\right] \geq \frac{1}{p(n)}$$

for infinitely many values of n, then there exists a polynomial $p'(\cdot)$ such that

$$\Pr\left[\begin{matrix} x_1, x_2, x_3 \text{are distinct and} \\ h(x_1) = h(x_2) = h(x_3) \end{matrix} \middle| \begin{matrix} h \leftarrow \mathcal{H}(1^n) \\ (x_1, x_2, x_3) \leftarrow M^{\mathcal{A},\mathcal{H}}(1^n, h) \end{matrix}\right] \geq \frac{1}{p'(n)}$$

for infinitely many values of n.

4 Multi-Collision-Resistant Function Families

A multi-collision-resistant hash function is a relaxation of standard collision-resistant hash function in which it is hard to find *multiple* collisions on the same value.

Definition 10 (Multi-Collision-Resistant Hashing). *Let $k = k(n)$ be a polynomial function. An efficient function family ensemble $\mathcal{H} = \{\mathcal{H}_n : \{0,1\}^{2n} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ is a (t,ϵ)-secure* k-multi-collision-resistant hash *(MCRH) function family if for any probabilistic algorithm $\mathcal{A}$ that runs in time at most $t(n)$, for large enough $n \in \mathbb{N}$:*

$$\Pr\left[\begin{array}{c} x_1,\ldots,x_k \text{ are distinct and} \\ h(x_1)=\cdots=h(x_k) \end{array} \middle| \begin{array}{c} h \leftarrow \mathcal{H}_n \\ (x_1,\ldots,x_k) \leftarrow \mathcal{A}(h) \end{array}\right] \leq \epsilon(n).$$

We call such $x_1,\ldots,x_k$ that map to the same value under h a k-wise collision. *Lastly, we say that $\mathcal{H}$ is a secure k-MCRH if it is $(p, 1/p)$-secure for every polynomial $p(\cdot)$.*

The Compression Ratio. In the definition above we assume that the hash function compresses its input from $2n$ bits into n, where the choice of the constant 2 is somewhat arbitrary. Our choice of linear compression rate (in contrast to, say, a polynomial compression rate) models the basic building blocks in most standards of cryptographic hash functions, such as the ones published by NIST (e.g., most of the SHA-x family).

When considering k-MCRH functions, a compression that eliminates less than $\log k$ bits is not of interest, since such a function exists unconditionally, say by chopping (there simply will be no k-wise collision).

The factor two compression is somewhat arbitrary as any k-MCRH that compresses $(1+\epsilon)n$ bits into n bits can be translated into a $(k^{1/\epsilon})$-MCRH that compresses $2n$ bits into n (e.g. via the Merkle-Damgård iterated construction [10, 42]). It is possible to assume an even stronger hash function that compresses by a polynomial factor, say n^2 bits into n bits (and this is sometimes useful; see the paragraph in the end of Sect. 6 for an example), but this is a strong assumption that we prefer to avoid.

For standard collision resistant hash function (with $k = 2$), it is known that composition allows to translate hash functions that compress by one bit into hash functions that compress by any polynomial factor (from n bits into n^δ bits for any constant $\delta > 0$). Obtaining a similar result for k-MCRH functions (with $k > 2$) without significantly compromising on the value of k in the resulting family is an open problem. In Sect. 6 we give a transformation in which the resulting family is $(k^{O(\log n)})$-MCRH.

Public vs. Private Coins. Our definition above is of a private-coin MCRH, namely, the coins used by the key-generation procedure are not given to the collision finder, but are rather kept secret. One can define the stronger public-coin variant in which the aforementioned coins are given to the attacker. The weaker notion is enough for our applications. There are (other) cases where this distinction matters, see Hsiao and Reyzin [31].

5 Tree Commitments from Multi-CRH

We show how to build a commitment scheme which is computationally-binding, statistically-hiding, round-efficient, has short commitments, and supports local

opening. We refer to Sect. 3.5 for the definition of a commitment scheme, computational-binding, statistical-hiding, and the efficiency measures of commitments we consider below.

Theorem 2. *Assume that there exists a (t, ϵ)-secure k-MCRH $\mathcal{H}$ for a polynomial $k = k(n)$ in which every function can be described using $\ell = \ell(n)$ bits. For any parameters $d = d(n)$ and $1 < z \leq n/2d$, there is commitment protocol for strings of length $2^d \cdot n$ with the following properties:*

1. *(t', ϵ')-computationally-binding for $\epsilon' = O\left(2^{\frac{d}{\log(n/(zd))}} \cdot \left(\frac{k^2}{2^{z-d}} + \epsilon\right)\right)$ and $t' = O\left(\frac{\epsilon'^2 \cdot t}{nk2^d \cdot p(n)}\right)$, where $p(\cdot)$ is some fixed polynomial function.*
2. *2^{-n}-statistically-hiding.*
3. *takes $O\left(\frac{d}{\log(n/(zd))}\right)$ rounds.*
4. *the commitment has length $O(d\ell + dn)$.*
5. *supports local opening of size $O(d^2 n)$.*

There are various ways to instantiate z compared to n and d, offering various trade-offs between security and efficiency. We focus here on the case in which we wish to commit on a polynomially-long string, that is, $d = c \log n$ for a constant $c \in \mathbb{N}$. In the following the big "O" notation hides constants that depend on c. Setting $z = n^{1-\delta}$ for a small constant $\delta > 0$, the parameters of our commitment scheme are:

1. $\epsilon' = O\left(\frac{k^2}{2^{n^{1-\delta}}} + \epsilon\right)$ and $t' = \frac{\epsilon'^2 \cdot t}{\mathsf{poly}(n)}$.
2. 2^{-n}-statistically-hiding.
3. takes $O(1)$ rounds.
4. the commitment has length $O(\ell + n \log n)$.
5. supports local opening of size $O(n \log^2 n)$.

This setting is very efficient in terms of rounds (it is a constant that depends solely on c) but suffers in security loss (the resulting scheme is at most $2^{-n^{1-\delta}}$-secure). In the regime where the MCRH is $(p, 1/p)$-secure for every polynomial p, our resulting scheme is as secure (i.e., $(p, 1/p)$-computationally-binding for every polynomial p).

In case ϵ is very small to begin with (e.g., much smaller than $2^{-n^{1-\delta}}$), we can use set z to be $z = n/(2c \log n)$ for a small constant $\delta > 0$. The resulting commitment scheme satisfies:

1. $\epsilon' = O\left(n^{c-1} \cdot \left(\frac{k^2}{2^{n/\log n}} + \epsilon\right)\right)$ and $t' = \frac{t \cdot \epsilon'^2}{\mathsf{poly}(n)}$.
2. 2^{-n}-statistically-hiding.
3. takes $O(\log n)$ rounds.
4. the commitment has length $O(\ell \log n + n \log n)$.
5. supports local opening of size $O(n \log^2 n)$.

This setting has a logarithmic number of rounds, but the security loss is much smaller than before (only of order $2^{-n/\log n}$).

Roadmap. Our protocol is constructed in two main steps. In the first step (given in Sect. 5.1) we construct a protocol with the above properties (i.e., Theorem 2) except that it is *not* statistically hiding (but is computationally-binding, takes few rounds, has short commitment, and supports local opening). In the second step (given in Sect. 5.2), we show how to *generically* bootstrap our commitment scheme, into one that is also statistically-hiding. This reduction is both efficient and security preserving with respect to all parameters.

5.1 A Computationally-Binding Scheme

The main ingredients in our first protocol are an MCRH (Definition 10) and a limited-independent family (Definition 2):

- A (t, ϵ)-secure k-MCRH for a polynomial $k = k(n)$:

$$\mathcal{H} = \{h \colon \{0,1\}^{2n} \to \{0,1\}^n\}.$$

  We assume that every $h \in \mathcal{H}$ can be described using $\ell = \ell(n)$ bits.
- A family of pairwise-independent functions mapping strings of length $2n$ to strings of length z:

$$\mathcal{G} = \{g \colon \{0,1\}^{2n} \to \{0,1\}^z\}.$$

  Recall that every $g \in \mathcal{G}$ can be described using $4n$ bits.

Description of the Protocol. Our protocol relies on the notion of a Merkle hash tree. This is a method to hash a long string into a short one using a hash function with fixed input length. Let $x \in \{0,1\}^\ell$ be a string. A Merkle hash tree is a binary tree T, associated with a string $x \in \{0,1\}^\ell$ and a hash function $h \colon \{0,1\}^{2n} \to \{0,1\}^n$. Let $x = x_1, \ldots, x_{2^d}$ be the decomposition of x into 2^d blocks, each of length n. Every node v in the tree has a sibling denoted by $N(v)$ (we assume that the sibling of the root is $\perp$). Every node v in the tree is labeled with a string $\pi_v \in \{0,1\}^n$. The tree has 2^d leaves $v_1, \ldots, v_{2^d}$ and the label of v_i are set to $\pi_{v_i} = x_i$. The labels of the rest of the nodes are computed iteratively from the leaves to the root. Given a node v whose both children u_1, u_2 are labeled with π_{u_1}, π_{v_2}, we set the label of v to be $\pi_v = h(\pi_{u_1}, \pi_{v_2})$. The node root has label y and we call it the root-hash.

Given a Merkle hash tree for a string $x = x_1 \ldots x_{2^d} \in \{0,1\}^{2^d \cdot n}$, let path_i be a set of nodes in the tree including the nodes on the path from x_i to the root of the tree and all their siblings along this path. We further let $P_i = \{\pi_v \mid v \in \mathsf{path}_i\}$ be the set of all the labels of the nodes in the set path_i (the labels of the nodes on the path from x_i to the root of the tree and the labels of their siblings). Each set path_i contains $2d$ nodes and thus the description size of each P_i is $2dn$ bits.

The commitment protocol $(\mathcal{S}, \mathcal{R}, \mathcal{V})$ specifies how to commit to a string of length $2^d n$. Our protocol uses a Merkle hash tree with a function h supplied by the receiver and a root hash replied by the sender. In the next round, the receiver chooses a limited-independence function g with z bits of output (z is

a tunable parameter) and sends it to the sender. The sender then computes g on the *hash values* of every pair of sibling nodes in the Merkle hash tree (i.e., $g(\pi_v \circ \pi_{N(v)})$). Then it concatenates all of these values into one long string s'. Then, the protocol continues in a recursive manner on this string s'. The length of s' is roughly $z2^d$ bits (there are 2^d internal nodes in the tree and each requires z bits) which is still too long to send as is, however is smaller than the original string s. This allows us to apply the same ideas recursively with the base case, committing on a string of length n, being the trivial protocol of sending the string as is. Choosing parameters carefully, we balance between the efficiency and security of our resulting commitment.

The commitment protocol for strings of length $2^d n$ for $d \geq 1$ is described in Fig. 1.

The commit protocol between $\mathcal{S}$ and $\mathcal{R}$

The sender $\mathcal{S}$ has string $s = s_1 \dots s_{2^d}$ where $s_i \in \{0,1\}^n$ for all $i \in [2^d]$.

1. $\mathcal{R} \Rightarrow \mathcal{S}$: Sample $h \leftarrow \mathcal{H}$ and send h.
2. $\mathcal{S} \Rightarrow \mathcal{R}$: Compute a Merkle hash-tree T of s using h and send the root-hash y. Let π_v be the hash value in the tree for node $v \in T$, and let $P_i = \left\{\pi_v \circ \pi_{N(v)}\right\}_{v \in \mathsf{path}_i}$ for all $i \in [d]$.
3. $\mathcal{R} \Rightarrow \mathcal{S}$: Sample $g \leftarrow \mathcal{G}$ and send g.
4. $\mathcal{S} \Leftrightarrow \mathcal{R}$: Recursively interact to commit on the string $s' = u_1 \circ \dots \circ u_{2^d}$, where $u_i = \left\{g(\pi_v \circ \pi_{N(v)})\right\}_{v \in \mathsf{path}_i}$. Notice that $|s'| = 2^d \cdot dz = 2^{d'} n$, where $d' = d - (\log n - \log z - \log d)$. Denote the outputs of the sender and receiver by

$$((D_1, \dots, D_{2^{d'} n}), C) \leftarrow \langle \mathcal{S}(1^n, s'), \mathcal{R} \rangle.$$

The output of each party

- $\mathcal{R}$'s output: The receiver $\mathcal{R}$ outputs $\mathsf{com} = (h, y, g, C)$.
- $\mathcal{S}$'s output: The sender $\mathcal{S}$ outputs $(\mathsf{decom}_1, \dots, \mathsf{decom}_{2^d})$, where decom_i is defined as follows: Let i' be the index in s' of the block containing u_i. Set $\mathsf{decom}_i = (s_i, P_i, D_{i'})$.[a]

The local-opening procedure $\mathcal{V}$

The verifier $\mathcal{V}$ has an index $i \in [2^d]$, a decommitment decom_i, and a commitment com.

1. Verify that s_i appears in P_i in the right location.
2. Verify that the values in P_i are consistent with respect to h and y.
3. Compute $u_i = \left\{g(\pi_v \circ \pi_{N(v)})\right\}_{v \in \mathsf{path}_i}$, where $P_i = \left\{\pi_v \circ \pi_{N(v)}\right\}_{v \in \mathsf{path}_i}$. Recursively compute $u_i' \leftarrow \mathcal{V}(i', D_{i'}, C)$ and verify that $u_i' = u_i$.
4. If all tests pass, output s_i. Otherwise, output $\perp$.

[a] We assume without loss of generality that each u_i is contained in a single block (otherwise, we pad the string).

Fig. 1. Our commitment protocol for strings of length $2^d \cdot n$.

Rounds and Communication Analysis. Denote by $\mathsf{Size}(2^d n)$, $\mathsf{Rounds}(2^d n)$, and $\mathsf{Decom}(2^d n)$, the *total size* of the commitment, the *number of rounds* of the commit stage, and the *size of a local opening* on a string of length $2^d n$, respectively. The commitment consists of a description of a hash function $h \in \mathcal{H}$ (whose size is denoted by $\ell(n)$), the root-hash value of a Merkle hash-tree (which is of size n), a function $g \in \mathcal{G}$ (which is of size $4n$), and the recursive commitment. The opening for an index $i \in [2^d]$ consists of a block (of size n), the full i-th path (which consists of $2dn$ bits), and the recursive opening.

Recall that the protocol for committing on strings of length $2^d n$ uses (recursively) a protocol for committing on strings of length $2^{d'} n$, where

$$d' = d - (\log n - \log z - \log d) = d - \log(n/(zd)).$$

Moreover, the commitment protocol on strings of length n has communication complexity n, consists of a single round and the opening is n bits. Thus, the total number of recursive call will be

$$\left\lceil \frac{d}{\log(n/(zd))} \right\rceil.$$

We get that the total number of communication rounds is bounded by

$$\mathsf{Rounds}(2^d n) \leq 3 + \mathsf{Rounds}(2^{d'} n) \leq \cdots \leq \left\lceil \frac{3d}{\log(n/(zd))} \right\rceil + O(1)$$

Actually, since each recursive call consists of three messages (except the base of the recursion), we can join the last round of every iteration with the first round in the next one. Therefore, we can get the improved bound

$$\mathsf{Rounds}(2^d n) \leq \frac{2d}{\log(n/(zd))} + O(1)$$

The size of a commitment is bounded by

$$\begin{aligned} \mathsf{Size}(2^d n) &\leq \ell(n) + 5n + \mathsf{Size}(2^{d'} n) \\ &\leq \left\lceil \frac{d \cdot (\ell(n) + 5n)}{\log(n/(zd))} \right\rceil \leq d \cdot (\ell(n) + 5n). \end{aligned}$$

The size of a local opening is bounded by

$$\begin{aligned} \mathsf{Decom}(2^d n) &\leq n + 2dn + \mathsf{Decom}(2^{d'} n) \\ &\leq \left\lceil \frac{3d^2 n}{\log(n/(zd))} \right\rceil \leq 3d^2 n. \end{aligned}$$

Computational-Binding. We show that our protocol is (t_d, ϵ_d)-computationally-binding for strings of length $2^d \cdot n$. We assume that the k-MCRH is (t, ϵ)-secure and that the internal protocol for strings of length $2^{d'} n$ is $(t_{d'}, \epsilon_{d'})$-computationally-binding. We set ϵ_d to satisfy the following recursive relation:

$$\epsilon_d = \frac{4k^2}{2^{z-d}} + 4\epsilon + 4\epsilon_{d'}.$$

Plugging in the number of rounds our recursion takes, we get that

$$\epsilon_d \leq 2^{\frac{6d}{\log n - \log z - \log d}} \cdot \left(\frac{4k^2}{2^{z-d}} + 4\epsilon \right).$$

Let $\mathcal{S}^*$ be a (cheating) adversary that runs in time t_d and breaks the binding of the protocol, namely, with probability ϵ_d, $\mathcal{S}^*$ is able to (locally) open the commitment in two ways without getting caught. That is, after the commitment stage, there is an index $i \in [2^d]$ for which the adversary $\mathcal{S}^*$ is able to open the corresponding block in two *different* ways with probability ϵ_d:

$$\Pr_{\mathcal{S}^*, \mathcal{R}} \left[\begin{array}{c} (i, \mathsf{decom}_i^0, \mathsf{decom}_i^1, \mathsf{com}) \leftarrow \langle \mathcal{S}^*(1^n), \mathcal{R} \rangle \text{ and} \\ \perp \neq \mathcal{V}(i, \mathsf{decom}_i^0, \mathsf{com}) \neq \mathcal{V}(i, \mathsf{decom}_i^1, \mathsf{com}) \neq \perp \end{array} \right] \geq \epsilon_d, \tag{1}$$

where the probability is taken over the random coins of both $\mathcal{S}^*$ and $\mathcal{R}$. The randomness of $\mathcal{R}$ consists of uniformly chosen functions $h \leftarrow \mathcal{H}$ and a function $g \leftarrow \mathcal{G}$, so we can rewrite Eq. (1) as

$$\Pr_{\substack{\mathcal{S}^*, h \leftarrow \mathcal{H}, \\ g \leftarrow \mathcal{G}}} \left[\begin{array}{c} (i, \mathsf{decom}_i^0, \mathsf{decom}_i^1, \mathsf{com}) \leftarrow \langle \mathcal{S}^*(1^n), (h, g) \rangle \\ \text{and} \perp \neq \mathcal{V}(i, \mathsf{decom}_i^0, \mathsf{com}) \neq \mathcal{V}(i, \mathsf{decom}_i^1, \mathsf{com}) \neq \perp \end{array} \right] \geq \epsilon_d. \tag{2}$$

We will show how to construct an adversary $\mathcal{A}$ that runs in time at most t and finds a k-wise collision relative to a randomly chosen hash function $h \leftarrow \mathcal{H}$. The procedure $\mathcal{A}$ will run the protocol $\langle \mathcal{S}^*(1^n), (h, g) \rangle$ with the given h and a function g chosen uniformly at random and get (with good probability) two valid and different openings for some index i. Then, $\mathcal{A}$ will *partially* rewind the adversary $\mathcal{S}^*$ to the stage after he received the hash function h and replied with the root hash y (of s), and execute it again but with a fresh function $g \leftarrow \mathcal{G}$. With noticeable probability, this will again result with two valid openings for some (possibly different) index i. Repeating this process enough times, $\mathcal{A}$ will translate a large number of different (yet valid) decommitments into a large number of collisions relative to h. The fact that these collisions are distinct (with good probability) will follow from the fact that the g's are sampled independently in every repetition.

We introduce some useful notation. Recall that $\mathcal{S}^*$ is the sender that is able to locally open its commitment in two different ways. We slightly abuse notation and also think of $\mathcal{S}^*$ as a distribution over senders (this is without loss of generality since we can assume it chooses all of its randomness ahead of time). For an index $i^* \in [R]$, string $y \in \{0,1\}^n$, we denote by $\mathcal{S}^*|_{h,y}$ a distribution over all senders $\mathcal{S}^*$ in which the first message received is h and the reply is the root-hash y. For a string y that is sampled by choosing $h \leftarrow \mathcal{H}$ and running $\mathcal{S}^*$ for one round to obtain y, the adversary $\mathcal{S}^*|_{h,y}$ is uniformly chosen from all possible continuations of the adversary $\mathcal{S}^*$ given these first two messages. Given this notation, we can write the adversary $\mathcal{S}^*$ as a pair of two distributions $(Y_h, \mathcal{S}^*|_{h,Y_h})$, where Y_h is the distribution of the first message $\mathcal{S}^*$ sends in response to h, and $\mathcal{S}^*|_{h,Y_h}$ is the distribution over the rest of the protocol.

In a high-level, our algorithm $\mathcal{A}$ maintains a binary tree of depth d and 2^d leaves, in which each node v is associated with a set of labels S_v. At the beginning, each such set S_v is initialized to be empty. The algorithm $\mathcal{A}$ uses $\mathcal{S}^*$ to get many valid pairs of openings decom_i^0 and decom_i^1 for an index i:

$$\mathsf{decom}_i^0 = (s_i^0, P_i^0, D_{i'}^0) \text{ and } \mathsf{decom}_i^1 = (s_i^1, P_i^1, D_{i'}^1)$$

that are consistent with a commitment:

$$\mathsf{com} = (h, y, g, C).$$

Since both openings are valid and $s_i \neq s_i'$, it must be that (1) s_i^0 (resp., s_i^1) appears in P_i^0 (resp., P_i^1) and (2) P_i^0 and P_i^1 are consistent with the root-hash y. Thus, it must be that $P_i^0 \neq P_i^1$ and there is a node v on the path path_i that is the first (going from the root to the leaves) non-trivial collision between P_i^0 and P_i^1. Now, by the induction hypothesis, the probability that $\mathcal{S}^*$ cheats in the internal decommitment is small, and since g is sampled uniformly at every iteration, we show that it must be a *new* collision that did not appear before. We will identify the location of the collision at a node v and add this collision to the set S_v. See Fig. 2 for a precise description of $\mathcal{A}$.

The adversary $\mathcal{A}(1^n, h)$:

1. For all nodes v, set $S_v = \emptyset$ to be the empty set.
2. Send to $\mathcal{S}^*$ the function h and receive a root-hash $y \in \{0,1\}^n$.
3. Do the following $T = 50nk2^d/\epsilon_d^2$ times:
   (a) Sample $g \leftarrow \mathcal{G}$.
   (b) Obtain $(i, \mathsf{decom}_i^0, \mathsf{decom}_i^1, \mathsf{com}) \leftarrow \langle \mathcal{S}^*|_{h,y}(1^n), g\rangle$.
   (c) Parse $\mathsf{decom}_i^0 = (s_i^0, P_i^0, D_{i'}^0)$ and $\mathsf{decom}_i^1 = (s_i^1, P_i^1, D_{i'}^1)$. Parse $\mathsf{com} = (h, y, g, C)$.
   (d) If any of the following occurs, continue to the next iteration:
      i. $\perp \neq \mathcal{V}(i, \mathsf{decom}_i^0, \mathsf{com}) \neq \mathcal{V}(i, \mathsf{decom}_i^1, \mathsf{com}) \neq \perp$.
      ii. $\perp \neq \mathcal{V}(i', D_{i'}^0, C) \neq \mathcal{V}(i', D_{i'}^1, C) \neq \perp$.
   (e) Let v the node of the first (from the root to the leaves) non-trivial collision between P_i^0 and P_i^1. Let $X = \pi_v^0, \pi_{N(v)}^0$ and $Y = \pi_v^1, \pi_{N(v)}^1$ be the values of the collision for P_i^0 and P_i^1, respectively. Add X and Y to S_v.
   (f) If there exists a node v for which $|S_v| \geq k$, then output S_v and halt.
4. Output $\perp$.

Fig. 2. The algorithm $\mathcal{A}$ to find a k-wise collision in $\mathcal{H}$.

The analysis of the adversary $\mathcal{A}$ can be found in the full version [38].

Remark 1 (Optimization I: recycling h). In the recursive step of our protocol, we can use the same hash function h as in the first step of the recursion. This saves sending its description (which is of size $\ell(n)$ at every iteration and the resulting size of a commitment is $(O(\ell(n) + d^2 n)$.

Remark 2 (Optimization II: almost-universal hashing). In our protocol we used a pairwise independent hash function whose description size is proportional to their input size. This costs us in communication. We could save communication by using almost-universal hash functions whose description size is proportional to their output size.

5.2 Getting Statistical-Hiding Generically

We show how to transform any short commitment scheme Π that is computationally binding (but perhaps not hiding) to a new scheme Π' that is short, computationally binding and *statistically hiding*. Moreover, if Π admits a local-opening, then Π' admits a local-opening as well. Our transformation is information theoretic, adds no additional assumptions and preserves the security, the number of rounds and communication complexity of the original scheme (up to a small constant factor).

High-Level Idea. The underlying idea of our transformation is to leverage the fact that the commitment protocol Π is short. Specifically, when committing to a long string $s \in \{0,1\}^{n^c}$ for some large $c \in \mathbb{N}$, the communication is very short: $\Lambda(n)$ bits for a fixed polynomial function.[13] Thus, when we commit to s, a large portion of s is not revealed to the receiver by the protocol so this part of s is statistically hidden. The question is how to make sure that all of s is hidden.

Our solution takes advantage of the fact that some fraction of s remains hidden. We commit on a random string r that is independent of s and slightly longer. Then, we extract from r the remaining randomness r' *given the communication of the protocol* using a strong extractor (see Sect. 3.2). Finally, we commit on the string $s \oplus r'$. We need to show that this scheme is computationally-binding and statistically-hiding. The former follows from the computational-binding of the original scheme. The latter follows from the fact that r' is completely hidden to the receiver and it masks the value of s.

The details of this transformation appear in the full version [38].

6 Four-Round Short Commitments from Multi-CRH

We show how to construct a *4-round* short commitment protocol based on a family of k-MCRH functions. Compared to the protocol from Sect. 5, this protocol has *no* local opening and is secure only for *constant* values of k. However, it consists only of 4 rounds. Furthermore, using techniques similar to Sect. 5.2 the protocol can be made also statistically-hiding which suffices for some applications such as statistical zero-knowledge arguments [8].

We discuss methods that allow us to prove security even for polynomial values of k towards the end of the section.

[13] Our protocol from Sect. 5.1 has an additional linear dependence on c, but we will ignore this in this section to simplify notation.

Theorem 3. *Assume that there exists a (t, ϵ)-secure k-MCRH $\mathcal{H}$ for a constant k in which every function can be described using $\ell = \ell(n)$ bits. For any $c \in \mathbb{N}$, there is a commitment protocol for strings of length n^c with the following properties:*

1. *(t', ϵ')-computationally-binding for $\epsilon' = 4\epsilon + \frac{O(k^{c \log n})}{2^n}$ and $t' = \frac{t \cdot \epsilon'^2}{O(k^{c \log n}) \cdot p(n)}$, where $p(\cdot)$ is some fixed polynomial function.*
2. *takes 4 rounds (i.e., 4 messages).*
3. *the commitment has length $\ell + O(n)$.*

Proof. We describe a commitment protocol for a string $s = s_1 \dots s_{2^d}$ of length $2^d \cdot n$ for $d = (c-1) \cdot \log n$. We show that our protocol is computationally-binding and getting statistical-hiding can be done using the generic transformation from Sect. 5.2. Our protocol uses a Merkle hash-tree as described in Sect. 5. The main observation made in this construction is that before using the Merkle hash-tree we can use a special type of encodings, called list-recoverable codes (see Definition 4) to get meaningful security. Let $C\colon \{0,1\}^{2^d n} \to (\{0,1\}^{2n})^{2^{d'}}$ be an (ℓ, L)-list-recoverable code for $\ell = k^d$, $L = O(k^d)$, and $d' = O(\log n)$ with some large enough hidden constants. Such a code exists by Theorem 1 (with efficient encoding and list-recovery). Let $\mathcal{G}$ be a family of (2^{-n})-almost pairwise-independent functions mapping strings of length $2^d n$ to strings of length n (see Definition 3): $\mathcal{G} = \{g\colon \{0,1\}^{2^d n} \to \{0,1\}^n\}$. Recall that every $g \in \mathcal{G}$ can be described using at most $2n + \log(2^d n) + \log(2^n) \leq 4n$ bits.

The commitment protocol for a string $s = s_1 \dots s_{2^d}$ of length $2^d \cdot n$ for $d = (c-1) \cdot \log n$ works as follows. The receiver first sends a description of an $h \leftarrow \mathcal{H}$ which is a k-MCRH to the sender. The sender then computes the encoding $C(s)$ of s and computes the Merkle hash tree of $s' = C(s)$ (and not of s). The sender sends the root of the hash tree y to the receiver. The receiver replies with a hash function $g \leftarrow \mathcal{G}$ and finally the sender replies with $u = g(s)$. The opening is done is the natural way by letting the sender reveal s to the receiver who then simulates the computation and makes sure that the messages y and u are consistent. See Fig. 3 for the precise description.

By the description of the protocol, one can see that the protocol consists of 4-rounds and has communication complexity of $\ell + n + 4n + n = \ell + 6n$ bits. In addition, for the honest sender the verification succeeds with probability 1.

The analysis of the commitment scheme can be found in the full version [38].

Supporting Arbitrary Larger k. The reason why we could prove security only for constant values of k stems from the fact that our adversary $\mathcal{A}$ for the MCRH runs in time proportional to k^d. The source of this term in the running time is that our Merkle hash tree in the construction is of depth $d = O(\log n)$ and when counting the number of possible openings per coordinate in a leaf, we get k^d possibilities. Our adversary $\mathcal{A}$ basically "collects" this number of different openings for some coordinate and thereby finds a k-wise collision. If k is super-constant the running time of $\mathcal{A}$ becomes super-polynomial.

The commit protocol between $\mathcal{S}$ and $\mathcal{R}$

The sender $\mathcal{S}$ has string $s = s_1 \dots s_{2^d}$ where $s_i \in \{0,1\}^n$ for all $i \in [2^d]$.

1. $\mathcal{R} \Rightarrow \mathcal{S}$: Samples $h \leftarrow \mathcal{H}$ and sends h.
2. $\mathcal{S} \Rightarrow \mathcal{R}$: Compute $s' = C(s)$ and a Merkle hash-tree T of s' using h and send the root-hash y.
3. $\mathcal{R} \Rightarrow \mathcal{S}$: Sample $g \leftarrow \mathcal{G}$ and send g.
4. $\mathcal{S} \Leftrightarrow \mathcal{R}$: Send $u = g(s)$ to the receiver

The output of the receive is $\mathsf{com} = (h, y, g, u)$**.**

The verifier $\mathcal{V}$ gets the input s simulates the sender to verify y and u.

Fig. 3. Our four-round commitment protocol for strings of length $2^d \cdot n$.

There are two paths we can take to bypass this. One is to assume super-polynomial security of the MCRH and allow our adversary to run in super-polynomial time. The second assumption is to assume that we start with a stronger MCRH that compresses by a polynomial factor (i.e., from $n^{1+\Omega(1)}$ to n) rather than by a factor of 2. This will cause the Merkle hash tree to be of constant depth. Under either of the assumptions, we can support polynomial values of k (in n). However, both assumptions are rather strong on the underlying MCRH and we thus prefer to avoid them; see Sect. 4 for a discussion.

Domain Extension of MCRH Functions. The construction we gave in the proof of Theorem 3 can be viewed as a domain extension method for MCRH functions. Specifically, given a k-MCRH f that maps $2n$ bits into n bits, we constructed a function g that maps $m = m(n)$ bits into n bits for any polynomial $m(\cdot)$ such that g is a $((k-1)^{\log m} + 1)$-MCRH.

Other Uses of List-Recoverable Codes for Domain-Extension. List-recoverable codes have been proven useful in various applications in cryptography. The work of Maurer and Tessaro [40] considers the problem of a extending the domain of a public random function. They show how to use a length-preserving random function f on n input bits, to get a variable-input-length function g that maps arbitrary polynomial-size inputs into arbitrary polynomial-size outputs such that g is indistinguishable from a random function for adversaries making up to $2^{(1-\epsilon)n}$ queries to g. One of their main components in the construction is a list-recoverable code (there referred to as an *input-restricting function family*).

Building on ideas and techniques of [40], Dodis and Steinberger [13] showed that list-recoverable codes are also useful for security preserving domain extension of message-authentication codes (with "beyond-birthday" security).

More recently, Haitner et al. [25] studied the possibility of a *fully parallel* (i.e., non-adaptive) domain-extension scheme that realizes a collision-resistant

hash function. Starting with a *random* function f that maps n bits into n bits, they construct a function g that maps $m(n)$ bits into n bits for any polynomial $m(\cdot)$, makes only parallel calls to f, and requiring an attacker to make at least $2^{n/2}$ queries to g to find a collision with high probability. Their construction uses list-recoverable codes and they show that such codes are actually necessary for this task.

7 Separating Multi-CRH from CRH

In this section we rule out fully black-box constructions (see Definition 9) of CRH functions from k-MCRH functions for $k > 2$. Our proof technique is inspired by the works of Asharov and Segev [2] and Haitner et al. [23], that are based in turn on ideas originating in the works of Simon [50] Gennaro et al. [14]) and Wee [54].

We present an oracle Γ relative to which there exists a 3-multi-collision-resistant hash function, but any collision-resistant hash function can be easily broken. The theorem below is stated and proved only for the case of standard CRH and 3-MCRH. The ideas in this proof naturally extend to a separation of k-MCRH from $(k+1)$-MCRH for all fixed values of k.

Theorem 4. *There is no fully black-box construction of a collision-resistant hash function family from a* 3*-multi-collision-resistant hash function family mapping* $2n$ *bits to* n *bits.*

The proof can be found in the full version [38].

8 UOWHFs, MCRHs and Short Commitments

We explore the relationship between short commitments, MCRHs and universal one-way hash functions (see Definition 6). Our main message is that short commitment protocols (with some requirements listed below) directly imply UOWHFs. The transformation is efficient in the sense that a description of a hash function corresponds to the messages sent by the receiver and evaluation of the function is done by executing the protocol. In some cases this gives a way to construct a UOWHF which is more efficient than the direct construction based on one-way functions or permutations [46,48] (see comparison below).

Theorem 5. *Any short commitment protocol in which the receiver is public-coin yields a universal one-way hash function with the following properties:*

1. *The key size is the total number of (public) coins sent by the receiver.*
2. *The length of inputs that the UOWHF supports is the length of messages the commitment protocol commits to and the length of the output is the amount of bits sent from the sender to the receiver.*
3. *The evaluation of a hash function amounts to a single execution of the commitment protocol.*

Plugging in our construction of short commitments from MCRH functions from Theorem 2, we obtain a new construction of a UOWHF for messages of length $n2^d$ starting with a k-MCRH for a polynomial $k = k(n)$. The key size in the resulting family is proportional to the number of bits sent from the receiver to the sender: $\ell + O(d \cdot n/\log n)$ bits, where ℓ is the size of an MCRH key.[14]

Using our construction of short commitments from MCRH functions from Theorem 3, we get a new construction of a UOWHF for messages of length $n2^d$ starting from a k-MCRH for any constant k. The key size in the resulting family is $\ell + O(n)$ bits. Notice that this term is independent of d and the hidden constant in the big "O" is pretty small.[15]

Comparison with Previous Constructions. Starting with a standard collision resistant hash function on short inputs, it is known how a collision resistant hash function on long inputs (based on the so called Merkle-Damgård iterated construction) [10,42]. This directly implies a UOWHF. The key size in the resulting construction is optimal: it is just a key for a single collision resistant hash. However, when starting with weaker building blocks the key size grows.

Naor and Yung [46] suggested a solution based on a tree hash (similar to a construction of Wegman and Carter for universal hash functions [55]). In their scheme, hashing a message of length $n2^d$ is done by a balanced tree such that in the i-th level $n2^{d-i}$ bits are hashed into $n2^{d-i-1}$ bits by applying the same basic compression function 2^{d-i-1} times. Each level in the tree requires its own basic compression function which results with a total of d keys for the basic UOWHF. Thus, the total size of a key in the resulting function is $d\ell$ bits, where ℓ is the bit size of a key in the basic UOWHF.

In case that the keys of the basic scheme (ℓ above) are rather long, Shoup [49], following on the XOR tree hash of Bellare and Rogaway [5], offered the following elegant scheme to transform a fixed-input UOWHF into a UOWHF that supports arbitrary long inputs. Given an input of 2^d blocks each of size n, for $i = 1, \ldots, 2^d$ we compute $c_i = h((c_{i-1} \oplus s_{\kappa(i)}) \circ m_i)$, where $s_0, \ldots, s_d$ are uniformly random "chaining variables", and $\kappa(i)$ chooses one of the elements $s_0, \ldots, s_d$.[16] Thus, the total size of a key in the resulting function is $\ell + (d+1)n$ bits.

The proof of Theorem 5 appears in the full version [38].

Acknowledgments. We are grateful to Noga Ron-Zewi for teaching us about list-recoverable codes, for multiple useful discussions, and for sharing with us a preliminary version of [28]. We greatly acknowledge Gilad Asharov and Gil Segev for educating us about black-box separations. We thank Iftach Haitner and Eran Omri for answering questions related to [25]. We also thank Stefano Tessaro for telling us about [13,40] and in particular for explaining the relation of [40] to this work.

[14] The overhead in the key size can be improved if the pairwise hash function is replaced by an almost uniform hash function as described in Remark 2.

[15] The concrete constant in our scheme is roughly 6 but we did not try to optimize it further.

[16] The function $\kappa(i)$ counts the number of times 2 divides i, that is, for $i \geq 1$, $\kappa(i)$ is the largest integer κ such that 2^κ divides i.

References

1. Alon, N., Goldreich, O., Håstad, J., Peralta, R.: Simple construction of almost k-wise independent random variables. Random Struct. Algorithms **3**(3), 289–304 (1992)
2. Asharov, G., Segev, G.: Limits on the power of indistinguishability obfuscation and functional encryption. SIAM J. Comput. **45**(6), 2117–2176 (2016)
3. Barak, B.: How to go beyond the black-box simulation barrier. In: 42nd Annual Symposium on Foundations of Computer Science, FOCS, pp. 106–115. IEEE Computer Society (2001)
4. Barak, B., Goldreich, O.: Universal arguments and their applications. SIAM J. Comput. **38**(5), 1661–1694 (2008)
5. Bellare, M., Rogaway, P.: Collision-resistant hashing: towards making UOWHFs practical. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052256
6. Berman, I., Degwekar, A., Rothblum, R.D., Vasudevan, P.N.: Multi collision resistant hash functions and their applications. IACR Cryptology ePrint Archive 2017, 489 (2017)
7. Bitansky, N., Kalai, Y.T., Paneth, O.: Multi-collision resistance: A paradigm for keyless hash functions. IACR Cryptology ePrint Archive 2017, 488 (2017)
8. Brassard, G., Chaum, D., Crépeau, C.: Minimum disclosure proofs of knowledge. J. Comput. Syst. Sci. **37**(2), 156–189 (1988)
9. Coppersmith, D.: Another birthday attack. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 14–17. Springer, Heidelberg (1986). https://doi.org/10.1007/3-540-39799-X_2
10. Damgård, I.B.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_39
11. Damgård, I., Pedersen, T.P., Pfitzmann, B.: On the existence of statistically hiding bit commitment schemes and fail-stop signatures. J. Cryptol. **10**(3), 163–194 (1997)
12. Damgård, I., Pedersen, T.P., Pfitzmann, B.: Statistical secrecy and multibit commitments. IEEE Trans. Inf. Theory **44**(3), 1143–1151 (1998)
13. Dodis, Y., Steinberger, J.: Domain extension for MACs beyond the birthday barrier. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 323–342. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_19
14. Gennaro, R., Gertner, Y., Katz, J., Trevisan, L.: Bounds on the efficiency of generic cryptographic constructions. SIAM J. Comput. **35**(1), 217–246 (2005)
15. Girault, M., Cohen, R., Campana, M.: A generalized birthday attack. In: Barstow, D., Brauer, W., Brinch Hansen, P., Gries, D., Luckham, D., Moler, C., Pnueli, A., Seegmüller, G., Stoer, J., Wirth, N., Günther, C.G. (eds.) EUROCRYPT 1988. LNCS, vol. 330, pp. 129–156. Springer, Heidelberg (1988). https://doi.org/10.1007/3-540-45961-8_12
16. Girault, M., Stern, J.: On the length of cryptographic hash-values used in identification schemes. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 202–215. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_21
17. Goldreich, O., Sahai, A., Vadhan, S.: Can statistical zero knowledge be made non-interactive? or on the relationship of SZK and *NISZK*. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 467–484. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_30

18. Guruswami, V., Indyk, P.: Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In: Proceedings on 34th Annual ACM Symposium on Theory of Computing, pp. 812–821. ACM (2002)
19. Guruswami, V., Indyk, P.: Linear time encodable and list decodable codes. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing, pp. 126–135. ACM (2003)
20. Guruswami, V., Indyk, P.: Linear-time list decoding in error-free settings. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 695–707. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27836-8_59
21. Guruswami, V., Sudan, M.: Improved decoding of Reed-Solomon and algebraic-geometry codes. IEEE Trans. Inf. Theory **45**(6), 1757–1767 (1999)
22. Guruswami, V., Umans, C., Vadhan, S.P.: Unbalanced expanders and randomness extractors from parvaresh-vardy codes. J. ACM **56**(4), 20:1–20:34 (2009)
23. Haitner, I., Hoch, J.J., Reingold, O., Segev, G.: Finding collisions in interactive protocols - tight lower bounds on the round and communication complexities of statistically hiding commitments. SIAM J. Comput. **44**(1), 193–242 (2015)
24. Haitner, I., Horvitz, O., Katz, J., Koo, C., Morselli, R., Shaltiel, R.: Reducing complexity assumptions for statistically-hiding commitment. J. Cryptol. **22**(3), 283–310 (2009)
25. Haitner, I., Ishai, Y., Omri, E., Shaltiel, R.: Parallel hashing via list recoverability. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 173–190. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_9
26. Haitner, I., Nguyen, M., Ong, S.J., Reingold, O., Vadhan, S.P.: Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. SIAM J. Comput. **39**(3), 1153–1218 (2009)
27. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. **28**, 1364–1396 (1999)
28. Hemenway, B., Ron-Zewi, N., Wootters, M.: Local list recovery of high-rate tensor codes & applications. In: 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS, pp. 204–215. IEEE Computer Society (2017)
29. Hemenway, B., Wootters, M.: Linear-time list recovery of high-rate expander codes. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) ICALP 2015. LNCS, vol. 9134, pp. 701–712. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47672-7_57
30. Hosoyamada, A., Sasaki, Y., Xagawa, K.: Quantum multicollision-finding algorithm. IACR Cryptology ePrint Archive 2017, 864 (2017)
31. Hsiao, C.-Y., Reyzin, L.: Finding collisions on a public road, or do secure hash functions need secret coins? In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 92–105. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_6
32. Impagliazzo, R.: A personal view of average-case complexity. In: Proceedings of the Tenth Annual Structure in Complexity Theory Conference, pp. 134–147. IEEE Computer Society (1995)
33. Impagliazzo, R., Levin, L.A., Luby, M.: Pseudo-random generation from one-way functions (extended abstracts). In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing, pp. 12–24. ACM (1989)
34. Impagliazzo, R., Luby, M.: One-way functions are essential for complexity based cryptography (extended abstract). In: 30th Annual Symposium on Foundations of Computer Science, FOCS, pp. 230–235. IEEE Computer Society (1989)

35. Joux, A.: Multicollisions in iterated hash functions. application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_19
36. Katz, J., Koo, C.: On constructing universal one-way hash functions from arbitrary one-way functions. IACR Cryptology ePrint Archive 2005, 328 (2005)
37. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: STOC, pp. 723–732. ACM (1992)
38. Komargodski, I., Naor, M., Yogev, E.: Collision resistant hashing for paranoids: Dealing with multiple collisions. IACR Cryptology ePrint Archive 2017, 486 (2017)
39. Komargodski, I., Naor, M., Yogev, E.: White-box vs. black-box complexity of search problems: ramsey and graph property testing. In: 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS, pp. 622–632 (2017)
40. Maurer, U., Tessaro, S.: Domain extension of public random functions: beyond the birthday barrier. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 187–204. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_11
41. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_21
42. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_40
43. Mironov, I.: Collision-resistant no more: hash-and-sign paradigm revisited. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 140–156. Springer, Heidelberg (2006). https://doi.org/10.1007/11745853_10
44. Naor, J., Naor, M.: Small-bias probability spaces: efficient constructions and applications. SIAM J. Comput. **22**(4), 838–856 (1993)
45. Naor, M., Ostrovsky, R., Venkatesan, R., Yung, M.: Perfect zero-knowledge arguments for NP using any one-way permutation. J. Cryptol. **11**(2), 87–108 (1998)
46. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing, pp. 33–43. ACM (1989)
47. Ngo, H.Q., Porat, E., Rudra, A.: Efficiently decodable compressed sensing by list-recoverable codes and recursion. In: 29th International Symposium on Theoretical Aspects of Computer Science, STACS. LIPIcs, vol. 14, pp. 230–241. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2012)
48. Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, pp. 387–394. ACM (1990)
49. Shoup, V.: A composition theorem for universal one-way hash functions. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 445–452. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_32
50. Simon, D.R.: Finding collisions on a one-way street: can secure hash functions be based on general assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054137
51. Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y.: The first collision for full SHA-1. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 570–596. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_19

52. Ta-Shma, A.: Explicit, almost optimal, epsilon-balanced codes. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC, pp. 238–251 (2017)
53. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_2
54. Wee, H.: One-way permutations, interactive hashing and statistically hiding commitments. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 419–433. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_23
55. Wegman, M.N., Carter, L.: New hash functions and their use in authentication and set equality. J. Comput. Syst. Sci. **22**(3), 265–279 (1981)