Blockcipher Based Hashing Revisited

Martijn Stam*

martijn.stam@epfl.ch
LACAL, EPFL, Switzerland

Abstract. We revisit the rate-1 blockcipher based hash functions as first studied by Preneel, Govaerts and Vandewalle (Crypto'93) and later extensively analysed by Black, Rogaway and Shrimpton (Crypto'02). We analyse a further generalization where any pre- and postprocessing is considered. This leads to a clearer understanding of the current classification of rate-1 blockcipher based schemes as introduced by Preneel et al. and refined by Black et al. In addition, we also gain insight in chopped, overloaded and supercharged compression functions. In the latter category we propose two compression functions based on a single call to a blockcipher whose collision resistance exceeds the birthday bound on the cipher's blocklength.

1 Introduction

The design and analysis of hash functions has recently come under renewed interest as a consequence of the NIST competition for a new hash function standard and the events that lead NIST to launch this competition in the first place. Of the many ideas to create a hash function, one of the oldest is to base it on a blockcipher, either explicitly such as in the Davies-Meyer construction or implicitly such as in SHA-1. The most common approach is to use a blockcipher taking n-bit blocks and n-bit keys and construct a $2n$-to-n bit compression function that makes only a single call to the blockcipher. Such schemes are called rate-1; in general the rate measures the number of message blocks that are hashed per call to the blockcipher.

Specific examples of rate-1 schemes were given by Davies-Meyer [25], Matyas-Meyer-Oseas [23], and Miyaguchi-Preneel [27, 31]. Preneel et al. [31] studied the general construction $H(M,V) = E(K,X) \oplus U$ where $K, X, U \in \{0, M, V, M \oplus V\}$ (or affine offsets thereof). They concluded that of the $4^3 = 64$ possibilities all but 12 allow collision attacks on the compression function with a complexity beating the birthday bound of $2^{n/2}$. Later Black et al. [8] showed that in the ideal cipher model these 12 compression functions are indeed collision resistant up to the birthday bound. More surprisingly, they also showed that an additional 8 construction are secure when properly iterated, even though collisions can easily be found in the respective compression functions. Duo and Li [14] later gave an alternative proof resulting in improved bounds.

Neither of these articles provides a deeper understanding of what makes these 12 respectively 8 schemes special to make them secure as compression function respectively as iterated hash function: what do they have in common that sets them apart from the other 44 schemes? Indeed, their respective security proofs still need to be rechecked, with minor modifications, for each of the schemes Despite these modifications being minor and fairly straightforward, this is not entirely satisfactory.

We isolate the properties that make Duo and Li's proof go through in the ideal cipher model for the collision resistance of rate-1 blockcipher based compression functions and their iterated hash functions. This sheds new light on what it is that provides the provable security for these schemes; indeed the classification by Black et al. can be *derived* from it. Central to our result is a more general type of compression function, consisting of the following three simple steps (see Figure 1):

1. Prepare key and plaintext: $(K, X) \leftarrow C^{\mathrm{PRE}}(M, V)$;
2. Make the call: $Y \leftarrow E(K, X)$;
3. Output the digest: $W \leftarrow C^{\mathrm{POST}}(M, V, Y)$.

* This work was partially funded by the European Commission through the ICT programme under Contract ICT-2007-216646 ECRYPT II. An extended abstract of this work appears in the proceedings of FSE'09. This is the full version.

Here E is a blockcipher (where key size $k = |K|$ and blocksize $n = |X| = |Y|$ may differ) and C^{PRE} and C^{POST} can be arbitrary functions given their respective domain and codomain. To avoid complications we will initially assume that input and output sizes of the compression function match those of the blockcipher, that is $m := |M| = k$ and $s := |V| = |W| = n$.

Similar to prior art we consider two types of schemes. Type-I schemes give rise to collision resistant compression functions whereas Type-II schemes give rise to compression functions that will turn into collision resistant hash functions when (Merkle-Damgård) iterated. Our taxonomy is slightly different from Black et al.'s: we allow schemes to be both Type-I and Type-II. Yet not all Type-I schemes are also of Type-II, even though a collision resistant compression function is well known to give rise to a collision resistant iterated hash function. This discrepancy stems from our definition of Type-I and Type-II schemes based on sufficient rather than necessary conditions.

Each type is defined by a set of three conditions on C^{PRE} and C^{POST}. Both types share the first two conditions and only differ in the third. The first condition is bijectivity of C^{PRE}, ensuring that each query to E (or its inverse) can only be used to evaluate the compression function for a single input. The second condition is that for all M, V the postprocessing $C^{\text{POST}}(M, V, \cdot)$ is bijective. This causes optimal transfer of unpredictabilibity of encryption answers to the output W. For Type-I schemes, the third condition is similar in nature to the second, making sure that the unpredictability of decryption answers carries over to the digest W as well. Formally, for all K, Y the modified postprocessing $C^{\text{POST}}(C^{-\text{PRE}}(K, \cdot), Y)$ should be bijective. For Type-II schemes, the third condition captures that for each decryption answer the corresponding input chaining variable V is highly unpredictable. Formally, for all K, the function $C^{-\text{PRE}}(K, \cdot)$ restricted to its second output V is bijective.

We provide a proof in the ideal cipher model that the probability of finding a collision in the compression function (for Type-I) respectively in the iterated hash function (for Type-II) is upper bounded by $\frac{1}{2}q(q-1)/(2^n - q)$, where q is the number of queries allowed to the adversary and n is the block size. For Type-I schemes (everywhere) preimage resistance is upper bounded by $q/(2^n - q)$. We also investigate the ramifications of our general classification for the classical PGV schemes. We conclude that the Type-I schemes are exactly those 12 identified before by Preneel et al. and later Black et al. Our Type-II schemes include the 8 schemes identified as Type-II by Black et al., plus an additional 8 schemes that were already known to be Type-I. Interestingly enough, Duo and Li have shown that the 8 schemes in the intersection of Type-I and Type-II are weaker than the 4 purely Type-I schemes with respect to second preimage resistance. (We have no explanation for this phenomenon.)

The benefits of our generalized framework become even clearer when analysing three more complex scenarios, when the restrictions on the parameters n, k, s, and m are being relaxed. Here we achieve the following results:

Chopped Compression Functions This corresponds to having an output size s of the compression function smaller than the blocksize n of the underlying blockcipher. A possible example is chopped Davies-Meyer; we show that, as one might expect, it is optimally collision resistant and preimage resistant. Note that chopping the output after each encryption frees up $n - s$ bits extra for message bits if we want to maintain $n + k = s + m$. In particular one can achieve compression even for fixed-key ($k = 0$) blockciphers. (This leads to a significant improvement of robustness [35] over the sponge construction [5, 6].)

Overloaded Compression Functions Here one tries to cram the compression function by having more input to the compression function than the blockcipher can handle, i.e., $s + m > n + k$. Examples are the sponge construction [5, 6] or the (related) compression function of Cubehash [4]: in both instances a fixed permutation is used ($k = 0$) yet the chaining variable is of blocksize ($s = n$). Our bound on collision resistance of the compression function is worse than if we would chop the chaining variable (to make space for the message). This superiority of a smaller chaining variable is somewhat similar to one reported by Stam [37], although for overloaded compression functions part of the problem are overly loose bounds. We also note that having a larger chaining variable gives the potential for better collision resistance in the iteration. The latter is still a wide open problem; we point out some of the challenges.

Supercharged Compression Functions The exact opposite of the previous two cases, since here one attempts to boost collision resistance beyond the birthday bound on the blocksize by setting $s > n$. Typically for such a scenario one expects more than one call to the blockcipher, however Lucks [22] recently gave a rate-1 double length compression function (based on a blockcipher with double key length) with collision resistance in the iteration close to the birthday bound. Stam [37] later gave a construction with similar collision resistance in the compression function, however he only considers non-adaptive adversaries with no access to any type of inverse oracle. We present a general framework for the collision resistance of compression functions in the ideal cipher model. In particular, we give a variant of Stam's construction collision resistant in the ideal cipher model (against adaptive adversaries). We also give a rate-$1/2$ compression function with collision resistance up to $2^{3n/4}$ queries based on a blockcipher with $k = n$ bit keys.

Although we consider the scenarios above representative, they are by no means exhaustive. For instance the SHA-3 candidate MD6 [32] employs a fixed permutation (so $k = 0$) with some input bits fixed, requiring $m+s < n$. Were collision resistance the only concern, this choice would be suboptimal: one should either increase m (and thereby the rate) or s (and thereby collision resistance). The benefit of their choice lies in the indifferentiability of the compression function that is achieved this way.

2 Background

2.1 General Notation

For a positive integer n, we write $\{0,1\}^n$ for the set of all bitstrings of length n. When X and Y are strings we write $X \,||\, Y$ to mean their concatenation and $X \oplus Y$ to mean their bitwise exclusive-or (xor). Unless specified otherwise, we will consider bitstrings as elements in the group $(\{0,1\}^n, \oplus)$.

For positive integers k and n, we let $\mathrm{Block}(k,n)$ denote the set of all blockciphers with k-bit key and operating on n-bit blocks. That is $E \in \mathrm{Block}(k,n)$ is a collection of 2^k permutations on the set $\{0,1\}^n$. Given that $E(K,\cdot)$ is a permutation for all $K \in \{0,1\}^k$, we write $D(K,\cdot)$ for its inverse.

Unless otherwise specified, all finite sets are equipped with a uniform distribution for random sampling. For example, $E \xleftarrow{\$} \mathrm{Block}(k,n)$ denotes random sampling from the set $\mathrm{Block}(k,n)$ and assignment to E.

We use the convention to write oracles that are provided to an algorithm as superscripts, for example $\mathcal{A}^E$ would be an adversary with black box access to some function E.

We write $B[Q;p]$ for the random variable counting the number of successes in Q independent Bernoulli trials, each with success probability p. It is binomially distributed, for integer $0 \le \kappa \le Q$:

$$\mathsf{Pr}\left[B[Q;p] = \kappa\right] = \binom{Q}{\kappa} p^{\kappa}(1-p)^{Q-\kappa} \ .$$

A Chernoff bound can be used to bound the tail probability for any $\kappa > Qp$, namely

$$\mathsf{Pr}\left[B[Q;p] > \kappa\right] < \left(\frac{epQ}{\kappa}\right)^{\kappa} \ .$$

2.2 Compression Functions and Hash Functions

A *compression function* is a mapping H from $\{0,1\}^m \times \{0,1\}^s$ to $\{0,1\}^s$ for some $m, s > 0$. A *blockcipher-based* compression function is a mapping $H : \{0,1\}^m \times \{0,1\}^s \to \{0,1\}^s$ given by a program that, given (M,V), computes $H^E(M,V)$ via access to an oracle $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ modeling an (ideal) blockcipher with k-bit key and operating on n-bit blocks.[1] A *single-call* blockcipher-based compression function calls its encryption oracle only once. Compression of a message block then proceeds as follows: Given an s-bit state V and m-bit message M, compute output $W = H^E(M,V)$ by

[1] One could also allow multiple blockciphers and use of decryption, but this scenario suffices for us.

1. Compute $(K, X) \leftarrow C^{\text{PRE}}(M, V)$.
2. Set $Y \leftarrow E(K, X)$.
3. Output $W \leftarrow C^{\text{POST}}(M, V, Y)$.

as illustrated by Figure 1. We will refer to $C^{\text{PRE}} : \{0,1\}^m \times \{0,1\}^s \to \{0,1\}^k \times \{0,1\}^n$ as preprocessing and to $C^{\text{POST}} : \{0,1\}^m \times \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^s$ as postprocessing.

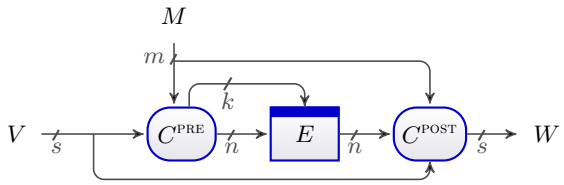

Fig. 1. General form of a $m + s$-to-s bit compression function based on a single call to the underlying blockcipher with k-bit key operating on n-bit block.

Since a blockcipher is easy to invert (given its key), an adversary trying to find for instance collisions will also have access to D. Figure 2 shows the effect of inverse queries on possible digests. To deal with inverse queries in our security analysis, we introduce the modified postprocessing $C^{\text{AUX}}(K, X, Y) = C^{\text{POST}}(C^{-\text{PRE}}(K, X), Y)$. In general, this is a function mapping triplets of strings to subsets of strings, since the result of $C^{-\text{PRE}}$ can have varying cardinality. For simplicity, when C^{PRE} is bijective, we understand C^{AUX} to have $\{0,1\}^n$ as its codomain.

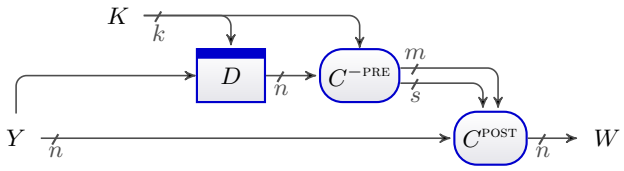

Fig. 2. The effect of inverse queries

A *hash function* is a mapping $\mathcal{H}$ from $\{0,1\}^*$ (the set of arbitrary length bitstrings) to $\{0,1\}^s$ for some $s > 0$. A compression function can be made into a hash function by iterating it. We briefly recall the standard Merkle-Damgård iteration [12, 26], where we assume that there is already some injective padding from $\{0,1\}^* \to (\{0,1\}^m)^* \backslash \emptyset$ in place (note that we disallow the empty message $\mathbf{M} = \emptyset$ as output of the injective padding). Given an initial vector $V_0 \in \{0,1\}^s$ define $\mathcal{H}^H : (\{0,1\}^m)^* \to \{0,1\}^s$ as follows for $\mathbf{M} = (M_1, \ldots, M_\ell)$ with $\ell > 0$:

1. Set $V_i \leftarrow H^E(M_i, V_{i-1})$ for $i = 1, \ldots, \ell$.
2. Output $\mathcal{H}^H(\mathbf{M}) = V_\ell$.

(Bearing this iteration in mind, given a compression function $H : \{0,1\}^m \times \{0,1\}^s \to \{0,1\}^s$ we will refer to the $\{0,1\}^m$ part of the input as 'message' and the $\{0,1\}^s$ part as the state or chaining variable.)

Collision Resistance. A *collision-finding adversary* is an algorithm with access to one or more oracles, whose goal it is to find collisions in some specified compression or hash function. It is standard practice to consider information-theoretic adversaries only. Currently this seems to provide the only handle to get any provable results. Information-theoretic adversaries are computationally unbounded and their

complexity is measured only by the number of queries made to their oracles. Without loss of generality, such adversaries are assumed not to repeat queries to oracles nor to query an oracle outside of its specified domain. We also assume that the adversary, before outputting a message, makes all calls necessary to evaluate the compressing function on that message. This does not decrease the advantage of the adversary, though it does increase its query complexity. Indeed, for some of the hash functions we will prove collision-resistant, it is in fact possible to find very long colliding messages using just a few queries; only honest evaluation of the hash function for those messages will increase an adversary's query complexity (see Appendix A for details). We stress that our proofs are in an idealized setting; hence as soon as the blockcipher is instantiated in practice weaknesses could surface and faster attacks might become feasible, even if the blockcipher itself has no weaknesses. Nonetheless, in all known instances of a security loss after instantiation, this either was the goal from the beginning (and the hash function is contrived) [7] or the blockcipher itself already has obvious shortcomings [30].

Despite the concept of initial vector being somewhat alien to a compression function on its own, it turns out helpful to consider a preimage to the initial vector a collision. Note that we deviate slightly from the definition given by Black et al. [8]. Whereas they fix the initial vector (and thus strictly speaking get a definition parametrized by the initial vector), we take the maximum over all possible initial vectors. Essentially Black et al.'s definition is finer grained, for instance in the preservation of collision resistance, their more refined result [8, Lemma 1] states that if the compression function is collision resistant with respect to a specific initial vector, then so is the iterated hash function with respect to that same initial vectors In contrast, our (weaker) statement (Theorem 2 gives that if the compression function is collision resistant with respect to all initial vectors, then so is the iterated hash function with respect to all initial vectors. The main reason we opted for the maximization approach is that most blockcipher based hash functions (in particular all those discussed in this paper) are equally secure for all initial vectors. In that case dropping the parametrization on the initial vector leads to both an easier and stronger statement.

Definition 1. *Let $n, k, m, s > 0$ be integer parameters. Let $H\colon \{0,1\}^m \times \{0,1\}^s \to \{0,1\}^s$ be a compression function taking oracle $E \in \mathrm{Block}(k,n)$. The collision-finding advantage of adversary $\mathcal{A}$ is defined to be*

$$\mathsf{Adv}^{\mathsf{coll}}_H(\mathcal{A}) = \max_{V_0 \in \{0,1\}^s} \Pr\Big[E \xleftarrow{\$} \mathrm{Block}(k,n), ((M,V),(M',V')) \leftarrow \mathcal{A}^{E,D}(V_0) : \\ (M,V) \neq (M',V') \text{ and } H^E(M,V) \in \{V_0, H^E(M',V')\}\Big] \ .$$

Define $\mathsf{Adv}^{\mathsf{coll}}_H(q)$ as the maximum advantage over all adversaries making at most q queries in total.

The quantity $\mathsf{Adv}^{\mathsf{coll}}_{\mathcal{H}}(q)$ denoting collision for the iterated hash function $\mathcal{H}^H$ is defined similarly: in this case the advantage of $\mathcal{A}$ is the maximum success probability taken over the choice of possible initial values V_0, which is input to $\mathcal{A}$. It is well known that the iterated hash function $\mathcal{H}$ is at least as secure as the compression function H it is based upon, as far as collision resistance is concerned [8, Lemma 1].

Theorem 2. *Let H be a blockcipher based compression function and let $\mathcal{H}$ be the iterated hash function based on H. Then*

$$\mathsf{Adv}^{\mathsf{coll}}_{\mathcal{H}}(q) \leq \mathsf{Adv}^{\mathsf{coll}}_H(q) \ .$$

Preimage Resistance. A *preimage-finding adversary* is an algorithm with access to one or more oracles, whose goal it is to find preimages in some specified compression function. There exist several definitions depending on the distribution of the element of which a preimage needs to be found. We opt for everywhere preimage resistance [33], which intuitively states that all points are hard to invert.[2] We also give the natural dual definition of somewhere preimage resistance [2], meaning that there is some point in the range that is hard to invert. This definition is typically to weak to use for any applications, but it often best captures a successful adversary's capabilities.

[2] However, contrary to the claim of Rogaway and Shrimpton [33], everywhere preimage resistance does not imply preimage resistance defined based on inverting the hash of a (sufficiently) random domain point [2].

Note that our particular definition of collision resistance for a compression function trivially implies everywhere preimage resistance. However, for everywhere preimage resistance we typically expect stronger bounds than those implied even by optimal collision resistance.

Definition 3. *Let $n, k, m, s > 0$ be integer parameters. Let $H\colon \{0,1\}^m \times \{0,1\}^s \to \{0,1\}^s$ be a compression function taking oracle $E \in \mathrm{Block}(k,n)$. The everywhere and somewhere preimage-finding advantages of adversary $\mathcal{A}$ are defined to be*

$$\mathsf{Adv}^{\mathsf{spre}}_H(\mathcal{A}) = \min_{W\in\{0,1\}^s} \Pr\left[E \xleftarrow{\$} \mathrm{Block}(k,n), (M',V') \leftarrow \mathcal{A}^{E,D}(W)\colon\ W = H^E(M',V')\right]$$

$$\mathsf{Adv}^{\mathsf{epre}}_H(\mathcal{A}) = \max_{W\in\{0,1\}^s} \Pr\left[E \xleftarrow{\$} \mathrm{Block}(k,n), (M',V') \leftarrow \mathcal{A}^{E,D}(W)\colon\ W = H^E(M',V')\right] .$$

Define $\mathsf{Adv}^{\mathsf{spre}}_H(q)$ and $\mathsf{Adv}^{\mathsf{epre}}_H(q)$ as the maximum advantage over all adversaries making at most q queries in total.

The quantities $\mathsf{Adv}^{\mathsf{spre}}_{\mathcal{H}}(q)$ and $\mathsf{Adv}^{\mathsf{epre}}_{\mathcal{H}}(q)$ denoting preimage resistance for the iterated hash function $\mathcal{H}^H$ are defined similarly (in this case the advantage of $\mathcal{A}$ is the maximum success probability taken over the choice of possible initial values V_0, which is input to $\mathcal{A}$). Everywhere preimage resistance is preserved in the (MD-)iteration [1], so we get:

Theorem 4. *Let H be a blockcipher based compression function and let $\mathcal{H}$ be the iterated hash function based on H. Then*

$$\mathsf{Adv}^{\mathsf{epre}}_{\mathcal{H}}(q) \le \mathsf{Adv}^{\mathsf{epre}}_H(q) \le \mathsf{Adv}^{\mathsf{coll}}_H(q) .$$

Adaptive Preimage Resistance and Preimage Awareness. Recently two new notions related to preimage resistance were introduced, namely adaptive preimage resistance [20] and preimage awareness [13]. Unlike the notions described above (which can easily be given for the standard model), it is specific to oracle-based functions (and no standard model equivalent is known). Indeed, both notions were introduced as a technical tool to prove indifferentiability of a hash function from a random oracle [24, 11], which is inherently outside the standard model. For simplicity, we will discuss only blockcipher based compression functions below. The generalization to hash functions or functions based on other primitives is straightforward; we refer to the original articles for the details and formalizations.

Recall that for a blockcipher based compression (or hash) function, one needs to make calls to the blockcipher to evaluate the function. Consequently, any given list of blockcipher queries (with their respective answers) determines a set of inputs for which the compression function can be evaluated, together with a set of corresponding 'reachable' digests. In a nutshell, an adversary breaks adaptive preimage resistance if it succeeds in making an as-yet unreachable digest reachable. That is, the adversary gets to make queries to its oracle, as well as the ability to commit to digests that are, at the time of commitment, unreachable given the current query history. It can make queries and commitments in arbitrary adaptive fashion. It succeeds if it finds a preimage for a digest it has previously committed to.

Preimage awareness [13] is defined slightly differently. Here the adversary has oracle access to both the underlying primitive (blockcipher) and an extractor. On input a range point, the extractor is tasked to return a preimage. The extractor does not have direct access to the underlying primitive, however it does get a transcript of the queries (including answers) that have been made by adversary so far. The adversary is successful if it can find an alternative preimage to the one returned by the extractor.

Several useful theorems about preimage awareness and adaptive preimage resistance are known. Preimage awareness is equivalent to a combination of collision resistance and adaptive preimage resisance. Moreover, preimage awareness is preserved by the Merkle-Damgård transform (equivalently, adaptive preimage resistance is preserved by the transform for collision resistant functions). Finally, applying an independent public random function (i.e., a fixed input length random oracle) to the output of a preimage aware function yields a new pseudorandom oracle (that is, indifferentiable from a true random oracle). This allows relatively easy construction of pseudorandom oracles with variable length input from a fixed length pseudorandom oracle applied to an iterated preimage aware compression function.

3 Classical Rate-1 Blockcipher Based Compression Functions

In this section we will deal with classical rate-1 blockcipher based compression functions, where the state size s equals the block length n of the blockcipher and the message size m matches the keysize k of the blockcipher. This includes the famous PGV hash functions [31]. (Note that for the PGV hash functions key length k and block length n are always equal as well; we do not pose this restriction.)

Following in the footsteps of Black et al. [8], we consider Type-I and Type-II compression functions. The former give optimal collision and preimage resistance in the compression function. The second type gives optimal collision resistance in the iteration; its preimage resistance can only be proved up to the birthday bound. One of the important differences with prior art is that we specify in very broad terms the requirements on C^{PRE} and C^{POST}. Essentially our primary concern here is for the proof to go through. In Section 3.3 we will discuss what our classification of Type-I and Type-II implies for the PGV hash functions.

The proof for Type-I schemes is fairly standard and straightforward. However, for the Type-II schemes we deviate from the one by Black et al. [8]. In particular, their proof is based upon colouring a directed graph where the vertices represent queries with all possible answers and arcs are drawn according to whether the input to one query is consistent with the output of the former, given the compression function under consideration. This leads to unwieldy graphs with a complicated notion of what consitutes a collision.

This counterintuitive use of graphs was fixed by Duo and Li [14] (as well as by Lucks [22]), who consider a directed graph where vertices correspond to chaining values and edges are drawn (or coloured) whenever a query has been made that would allow to move from one chaining value to the next. (Note that despite the directed nature of his graph, given an arc it is still unclear whether it was the result of a forward or an inverse query.) Moreover, for the actual bounding of collision resistance Duo and Li dispense with the direction of the arcs (that thus become edges). Although this seemingly aids the adversary (certain patterns in the graph will be deemed a success even when the underlying event on the hash function is not), this simplification leads to a tighter bound for the Type-II schemes, mainly because there is no longer any need to distinguish between several cases (whose success probability are subsequently added). Our proof (of Theorem 9) closely follows that of Duo and Li.

Note that even for Type-I schemes our bound appears a bit tighter than the one by Black et al., which is due to their simplification based on the the inequality $2(2^n - q) > 2^n$, at least for $q < 2^{n-1}$ (and for larger q most of the bounds become vacuous anyway). We believe the choice between tightness and simplicity in this case is one mainly of taste; we have opted for the former.

3.1 Type-I: Collision Resistant Compression Functions

Definition 5. *A single call blockcipher based compression function H^E is called rate-1 Type-I iff $n = s, k = m$ and the following three hold:*

1. *The preprocessing C^{PRE} is bijective.*
2. *For all M, V the postprocessing $C^{\text{POST}}(M, V, \cdot)$ is bijective.*
3. *For all K, Y the modified postprocessing $C^{\text{AUX}}(K, \cdot, Y)$ is bijective.*

Theorem 6. *Let H^E be a rate-1 Type-I compression function (based on a blockcipher with block size n). Then the advantage of an adversary in finding a collision in H^E after q queries can be upper bounded by*

$$\mathsf{Adv}^{\mathsf{coll}}_H(q) \leq \frac{1}{2} q(q+1)/(2^n - q) \; .$$

Proof. Let $V_0 \in \{0,1\}^n$ be given. A collision consists of two pairs (M, V) and (M', V') satisfying $H^E(M, V) = \{V_0, H^E(M', V')\}$ yet $(M, V) \neq (M', V')$. We will maintain a list of triples (M, V, W) such that $W = H^E(M, V)$ and the adversary has made the relevant queries to E and/or D. The list is initialized with $(-, -, V_0)$. Since we require the adversary to have made all relevant queries when

outputting a collision, we can upper bound the success probability of the adversary by bounding the probability of a collision occuring in this list. We show that any query, be it forward or inverse, will add at most one triple (M, V, W) to this list of computable compression functions, moreover the value W is almost completely out of the adversary's control.

Consider a forward query (K, X). By bijectivity of C^{PRE}, there is a unique pair (M, V) corresponding to this query. Thus, each forward query will add one triple (M, V, W) to the adversary's list of computable values. Since $C^{\text{POST}}(M, V, \cdot)$ is bijective for all M, V, the distribution of compression function output W is closely related to that of blockcipher output Y, which is close to being uniform. More precisely, suppose that so far t queries to E (and D) have been made involving key K, resulting in t plaintext-ciphertext pairs (X_i, Y_i) with $Y_i = E(K, X_i)$ for $i = 1, \ldots, t$. The answer to a fresh query to $E(K, \cdot)$ will therefore be $Y^* \neq Y_i, i = 1, \ldots, t$. Moreover, each of the $2^n - t$ answers is equally likely if E is an ideal cipher. Each possible answer Y^* will combine under C^{POST} with the pair (M, V) consistent with the (K, X) query being made, leading to a possible compression function outcome W^*. Because C^{POST} is bijective when (M, V) are fixed, distinct Y^* lead to distinct W^*, so there are $2^n - t$ possible outcomes W^*, all equally likely.

Similarly, consider an inverse query (K, Y). This yields a unique X and hence by bijectivity of C^{PRE}, there is a unique pair (M, V) corresponding to this query once answered. Thus, each inverse query will add one triple (M, V, W) to the adversary's list of computable values. This time bijectivity of $C^{\text{AUX}}(K, \cdot, Y)$ implies that the distribution of W is closely related to the (almost uniform) output distribution of D. Indeed, suppose that so far t queries to E have been made involving key K, resulting in t plaintext-ciphertext pairs (X_i, Y_i) with $Y_i = E(K, X_i)$ for $i = 1, \ldots, t$. The answer to a fresh query to $D(K, \cdot)$ will therefore be $X^* \neq X_i, i = 1, \ldots, t$. Moreover, each of the $2^n - t$ answers is equally likely if E is an ideal cipher. Each possible answer X^* will combine under $C^{-\text{PRE}}$ and C^{POST} with K and Y to a triple (M, V, W). Because for all K and Y the mapping from X to W is bijective (by assumption on C^{AUX}), distinct X^* lead to distinct W^*, so there are $2^n - t$ possible outcomes W^*, all equally likely.

As a result, after $i - 1$ queries the list of computable values contains i triples (M, V, W). The i'th query will add one triple with W uniform over a set of size at least $2^n - i + 1$. Thus the probability that the i'th query causes a collision with any of these triples is at most $i/(2^n - i + 1)$. Using a union bound, the probability of a collision after q queries can then be upper bounded by $\sum_{i=1}^{q} i/(2^n - i + 1) \leq \frac{1}{2}q(q+1)/(2^n - q)$. □

Theorem 7. *Let H^E be a rate-1 Type-I compression function (based on a a blockcipher with block size n). Then the advantage of an adversary in finding a preimage in H^E after q queries can be upper bounded by*

$$\mathsf{Adv}_H^{\mathsf{epre}}(q) \leq q/(2^n - q) \; .$$

Proof. Let $\mathcal{A}$ be an adversary that tries to find a preimage for its input σ. Assume that $\mathcal{A}$ asks its oracles E and D a total of q queries.

We recall the proof of Theorem 6, where we show that after $i - 1$ queries (to E or D) the list of computable values $W = H^E(M, V)$ contains $i - 1$ triples[3] (M, V, W). The i'th query will add one triple with W uniform over a set of size at least $2^n - i + 1$. Thus the probability that the i'th query hits σ is at most $1/(2^n - i + 1)$. Using a union bound, the probability of finding a preimage for σ after q queries can then be upper bounded by $\sum_{i=1}^{q} 1/(2^n - i + 1) \leq q/(2^n - q)$. □

3.2 Type-II: Collision Resistance in the Iteration

Definition 8. *A single call blockcipher based compression function H^E is called rate-1 Type-II iff $n = s, k = m$, and the following three hold:*

1. *The preprocessing C^{PRE} is bijective.*
2. *For all M, V the postprocessing $C^{\text{POST}}(M, V, \cdot)$ is bijective.*

[3] For preimage resistance we do not have to take $(-, -, V_0)$ into account.

3. For all K, $C^{-\mathrm{PRE}}(K,\cdot)$ restricted to V, its second output, is bijective.

Theorem 9. *Let H^E be a rate-1 Type-II compression function. If E is an ideal cipher with block size n, then the advantage of an adversary in finding a collision in the iterated hash function $\mathcal{H}^H$ after q queries is upper bounded by*

$$\mathsf{Adv}^{\mathsf{coll}}_{\mathcal{H}}(q) \leq \frac{1}{2}q(q+1)/(2^n-q)\ .$$

Proof. Let $V_0 \in \{0,1\}^n$ be $\mathcal{H}$'s initial vector.

We define an undirected graph $G = (V_G, E_G)$ with vertex set $V_G = \{0,1\}^n$—corresponding to all 2^n possible chaining values—and initially an empty edge set $E_G = \emptyset$. We will dynamically add edges based on the queries to E and D. In particular, we add an edge (V, W), labelled by M, if we know a message M such that $W = H^E(M,V)$ (or $V = H^E(M,V)$) and the relevant query to either E or D has been made. We claim that to find a collision would require constructing a ρ-shape containing the initial vector V_0. Suppose that $\mathcal{H}(\mathbf{M}) = \mathcal{H}(\mathbf{M}')$ with $\mathbf{M} \neq \mathbf{M}'$. Write $\mathbf{M} = (M_1, \ldots, M_\ell)$ and $\mathbf{M}' = (M'_1, \ldots, M'_{\ell'})$ and correspondingly $V_0, \ldots, V_\ell$ respectively $V'_0, \ldots, V'_{\ell'}$ for the chaining values of the iterated hash. Note that $V_0 = V'_0$ and $V_\ell = V'_{\ell'}$. Assume $\ell \leq \ell'$. Because $\mathbf{M} \neq \mathbf{M}'$, there exists a t such that $M_i = M'_i$ for all $0 \leq i < t$ but $M_t \neq M'_t$ (or possibly $\ell < t \leq \ell'$). As a result, the paths $(V_0, \ldots, V_t)$ and $(V'_0, \ldots, V'_t)$ are identical, but the edges (V_t, V_{t+1}) and (V'_t, V'_{t+1}) are distinct, even when V'_{t+1} happens to equal V_{t+1} (in particular, the edges are labelled differently). Since $V_\ell = V'_{\ell'}$ at some point the paths need to come together again, completing the ρ-shape. Note that due to our use of an undirected graph not every ρ-shape will lead to a collision though.

Since we are dynamically adding edges to the graph, components in the graph will also grow dynamically. Let T be the set of all nodes that are in a component containing a cycle or the initial vector V_0. The first claim is that after i queries, the set T has cardinality at most $i+1$. Indeed, the component containing V_0 has at most $i'+1$ nodes when i' edges are used. A cyclic component based on i' edges has at most i' nodes. Thus the initial vector component is the only component in T that causes the number of nodes larger than the number of edges, by at most one. Bijectivity of C^{PRE} implies that a query (either forward or inverse) will add at most one edge to the graph, so after i queries, there are at most i edges in the entire graph and at most $i+1$ nodes in T.

The second claim is that to complete a ρ-shape, either a cycle has to be completed within the V_0-component, or the V_0-component needs to be connected with a cycle. Either way, an edge has to be found of which both nodes are already part of T. The probability that on the i'th query a collision is found by a forward query is at most $i/(2^n-i)$: bijectivity of $C^{\mathrm{POST}}(M,V,\cdot)$ ensures that W is uniformly distributed over a set of size at least $2^n - i$, so hitting a set of size i occurs at most with said probability. Similarly, for an inverse query the probability of finding a collision on the i'th query using an inverse query is at most $i/(2^n-i)$: this time bijectivity of $C^{\mathrm{AUX}}(K,\cdot,Y)$ ensures that V is uniformly distributed over a set of size at least $2^n - i$.

We can now wrap up and conclude that the probability of finding a collision on the i'th query is at most $i/(2^n-i)$ and the probability after q queries is at most $\sum_{i=1}^{q} i/(2^n-i) \leq \frac{1}{2}q(q+1)/(2^n-q)$. □

3.3 Implications to the PGV Schemes

In this section we investigate how the 64 PGV schemes [31] fit in the general Type-I and Type-II framework. Recall that for the PGV-style schemes the blockcipher has key size equal to the block length; the compression function will look like $H^E(M,V) = E(K,X) \oplus U$ where $K, X, U \in \{C, M, V, M \oplus V\}$ and C is some fixed, publicly known bitstring. These restrictions can also be expressed in terms of C^{PRE} and C^{POST}. Our results are in line with the classification of Black et al. [8] and the tighter bounds by Duo and Li [14].

Let us first set up some notation. As is customary [17] for schemes with linear processing C^{PRE} and C^{POST}, we will represent the linear PGV schemes using matrices. We will use $\mathbb{Z}_2^2$ to express the way K,X, and U are functions of M and V: a vector $\mathbf{X} \in \mathbb{Z}_2^2$ corresponds to $X = \mathbf{X} \cdot \binom{M}{V}$, making a distinction between the linear map $\mathbf{X} \in \mathbb{Z}_2^2$ and the value $X \in \{0,1\}^n$. We will also write $\mathbf{X} =$

(X_M, X_V). For instance $\mathbf{X} = (10)$ has $X_M = 1$ and $X_V = 0$, corresponding to $X \leftarrow M$. We can safely ignore any affine part, so $\mathbf{U} = (00)$ can be thought of to correspond to the aforementioned $U \leftarrow C$. (This is without loss of generality, since translation by a constant will not affect bijectivity in either of the criteria used in Definitions 5 and 8.) Since there are 4 elements in $\mathbb{Z}_2^2$ and we have to pick 3 ($\mathbf{K}$,$\mathbf{X}$, and $\mathbf{U}$), there are 64 constructions to consider in total, corresponding to the 64 PGV schemes.

Observe that $\binom{K}{X} \leftarrow C^{\text{PRE}}(M,V) = \binom{\mathbf{K}}{\mathbf{X}}\binom{M}{V}$, $W \leftarrow C^{\text{POST}}(M,V,Y) = Y \oplus \mathbf{U}\binom{M}{V}$ and finally $W \leftarrow C^{\text{AUX}}(K,X,Y) = Y \oplus \mathbf{U}\binom{\mathbf{K}}{\mathbf{X}}^{-1}\binom{K}{X}$. For future reference, for invertible matrices $\binom{\mathbf{K}}{\mathbf{X}} \in \mathbb{Z}_2^{2\times 2}$ it holds

$$\begin{pmatrix} K_M & K_V \\ X_M & X_V \end{pmatrix}^{-1} = \begin{pmatrix} X_V & K_V \\ X_M & K_M \end{pmatrix} .$$

We are now ready to see what the requirements from Definitions 5 and 8 mean in terms of the vectors $\mathbf{K}$, $\mathbf{X}$ and $\mathbf{U}$ and hence for the classification and security of the PGV schemes. The 20 interesting schemes are listed in Table 1, where we have also included the $\imath$-indices assigned to these schemes by Black et al. [8]. When we write $H_\imath$ resp. $\mathcal{H}_\imath$ for $\imath \in \{1,\ldots,20\}$ we refer to this enumeration.

Lemma 10. *(The Type-I PGV schemes) A PGV scheme is Type-I iff $\binom{\mathbf{K}}{\mathbf{X}}$ and $\binom{\mathbf{K}}{\mathbf{U}}$ are both invertible matrices. In particular, $H_{1..12}$ are Type-I schemes.*

Proof. We will check the three conditions listed in Definition 5. Firstly, C^{PRE} needs to be bijective. This is equivalent to $\binom{\mathbf{K}}{\mathbf{X}}$ being invertible. Secondly, for all M, V the postprocessing $C^{\text{POST}}(M,V,\cdot)$ should be bijective. This is always the case for the schemes at hand, as can be easily verified. Finally, for all K, Y the modified postprocessing $C^{\text{AUX}}(K,\cdot,Y)$ should be bijective. For this we require $Y \oplus \mathbf{U}\binom{\mathbf{K}}{\mathbf{X}}^{-1}\binom{K}{X}$ to be bijective as a function of X for all Y and K. This simplifies to $(U_M K_V \oplus U_V K_M)X$ being bijective, or alternatively $(U_M K_V \oplus U_V K_M) = \det\binom{\mathbf{K}}{\mathbf{U}} = 1$. This is equivalent to stating that $\binom{\mathbf{K}}{\mathbf{U}}$ is invertible or that $\mathbf{U}$ is not in the span of $\mathbf{K}$ (given that $\mathbf{K} \neq (00)$ for bijective C^{PRE}).

Invertibility of $\binom{\mathbf{K}}{\mathbf{X}}$ reduces the choice of $\mathbf{K}$ and $\mathbf{X}$ from 16 to 6. For a given invertible $\binom{\mathbf{K}}{\mathbf{X}}$ exactly two values are possible for $\mathbf{U}$ (since (00) and $\mathbf{K}$ are ruled out), yielding $6 \cdot 2 = 12$ schemes in total. These are exactly the 12 schemes that PGV singled out as secure. □

We note that the requirements on the matrices $\binom{\mathbf{K}}{\mathbf{X}}$ and $\binom{\mathbf{K}}{\mathbf{U}}$ are similar to those given by Hirose [16] for the case of collision resistant double length compression functions based on two calls to a blockcipher with key size $k = 2n+1$ (where the $+1$ is used for domain separation purposes only). Indeed, our work can be extended [28] to the relevant 2-call scenario and thus derive the very requirements given by Hirose.

The requirements for the Type-II schemes turn out surprisingly simple: indeed apart from the preprocessing having full rank, the only requirement is that the key depends on the message. Consequently we end up with 16 Type-II schemes as opposed to only 8 given by Black et al. The 'additional' 8 schemes we identify are also Type-I, which explains why previously they were not classified as Type-II. Our results therefore suggest a subdivision of the PGV Type-I schemes, namely those that are also Type-II (being those with a key depending on the message) and those that are just Type-I (those whose key equals the chaining variable). The same subdivision was made by Duo and Li [14] in the context of second preimage resistance.

Lemma 11. *(The Type-II PGV schemes) A PGV scheme is Type-II iff $\binom{\mathbf{K}}{\mathbf{X}}$ is an invertible matrix with $K_M = 1$. In particular, $\mathcal{H}_{5..20}$ are Type-II schemes.*

Proof. We will check the three conditions listed in Definition 8. Firstly, C^{PRE} needs to be bijective. This is equivalent to $\binom{\mathbf{K}}{\mathbf{X}}$ being invertible. Secondly, for all M, V the postprocessing $C^{\text{POST}}(M,V,\cdot)$ should be bijective. This is always the case for the schemes at hand, as can be easily verified. Finally, for all K, $C^{-\text{PRE}}(K,\cdot)$ restricted to V, its second output, should be bijective. We need for all K that $\binom{M}{V} \leftarrow \binom{\mathbf{K}}{\mathbf{X}}^{-1}\binom{K}{X}$ is bijective as a function from X to V. This is true iff K_M equals 1, so $\mathbf{K} = (11)$ or $\mathbf{K} = (10)$. Each of these two choices of $\mathbf{K}$ comes with two possible choices of $\mathbf{X}$ to make $\binom{\mathbf{K}}{\mathbf{X}}$

$\binom{\mathbf{k}}{\mathbf{x}}\backslash\mathbf{s}$	(00)	(01)	(10)	(11)
$\binom{0\ 1}{1\ 0}$	insecure	insecure	$E_V(M) \oplus M^{1}$	$E_V(M) \oplus M \oplus V^{3}$
$\binom{0\ 1}{1\ 1}$	insecure	insecure	$E_V(M \oplus V) \oplus M^{4}$	$E_V(M \oplus V) \oplus M \oplus V^{2}$
$\binom{1\ 0}{0\ 1}$	$E_M(V)^{15}$	$E_M(V) \oplus V^{5}$	$E_M(V) \oplus M^{17}$	$E_M(V) \oplus M \oplus V^{7}$
$\binom{1\ 0}{1\ 1}$	$E_M(M \oplus V)^{19}$	$E_M(M \oplus V) \oplus V^{8}$	$E_M(M \oplus V) \oplus M^{20}$	$E_M(M \oplus V) \oplus M \oplus V^{6}$
$\binom{1\ 1}{0\ 1}$	$E_{M\oplus V}(V)^{16}$	$E_{M\oplus V}(V) \oplus V^{10}$	$E_{M\oplus V}(V) \oplus M^{12}$	$E_{M\oplus V}(V) \oplus M \oplus V^{18}$
$\binom{1\ 1}{1\ 0}$	$E_{M\oplus V}(M)^{13}$	$E_{M\oplus V}(M) \oplus V^{11}$	$E_{M\oplus V}(M) \oplus M^{9}$	$E_{M\oplus V}(M) \oplus M \oplus V^{14}$

Table 1. The 20 Secure PGV-style schemes, writing $E_K(X)$ for $E(K, X)$. Superscripted are the ι-indices from [8, Fig. 1 and 2]

invertible. There are four choices for $\mathbf{U}$ for a given pair $(\mathbf{K}, \mathbf{X})$, however, two of the choices for $\mathbf{U}$ are already covered by the Type-I schemes, so we only get two new cases, namely $\mathbf{U} = (00)$ and $\mathbf{U} = \mathbf{K}$. All in all 8 schemes are exclusively Type-II. □

Combining Lemmas 10 and 11 with Theorems 6, 7, and 9 then yields Corollary 12 below. For completeness [8, 31], it is known that the given upper bounds on the advantages are tight up to a small constant factor. Moreover, for $\mathcal{H}_{13..20}$ preimage resistance is worse than desired, namely $\mathsf{Adv}^{\mathsf{epre}}_{\mathcal{H}}(q) = \Theta(q^2/2^n)$ (due to a meet-in-the-middle attack). The remaining 44 PGV schemes do not offer any collision resistance in the iteration.

Corollary 12. *(Security of the PGV schemes) For $H_{1..12}$ it holds that $\mathsf{Adv}^{\mathsf{coll}}_{H}(q) \leq \frac{1}{2}q(q+1)/(2^n - q)$, and $\mathsf{Adv}^{\mathsf{epre}}_{H}(q) \leq q/(2^n - q)$; for $\mathcal{H}_{13..20}$ it holds that $\mathsf{Adv}^{\mathsf{coll}}_{\mathcal{H}}(q) \leq \frac{1}{2}q(q+1)/(2^n - q)$.*

4 Generalized Single Call Compression Functions

In the previous section we discussed the standard (single call) case where the input and output sizes of the compression function neatly matched those of the underlying blockcipher, in particular $m = k$ and $s = n$. In this section we let go of these restrictions and consider three more general scenarios.

First we will consider what could be called chopping the output of the compression (or really the scenario where $s < n$). For instance, the Davies-Meyer construction is optimally collision and preimage resistant, but what happens if you chop the output: is the security still optimal given the new output length (it is). A welcome benefit of chopping the output is that it frees up bits for the message. More precisely, if $s < n$ then we can have a larger m while maintaining $m + s = n + k$. In particular, compression becomes feasible even for fixed permutations (corresponding to $k = 0$). In view of the recent availability of huge size permutations constructions with $s < n$ gain traction; an example is Grindahl[18]. We will refer to this scenario as compression in the postprocessing, the corresponding H^E's are called chopped compression functions.

Similarly, one might also try to improve efficiency by squeezing in more bits of input in the compression function than can be input to the primitive (this corresponds to $m + s > n + k$). We call this compression in the preprocessing and speak of overloaded compression functions. Like the previous scenario, this opens up the possibility of achieving compression based on a single fixed permutation. We suggest a general Type-I compression function and give a bound on its collision resistance and preimage resistance. Security in the iteration is more complicated here: we discuss related work and point out some challenging open problems.

Finally we deal with the problem of getting security beyond the block length of the blockcipher, that is $s > n$. Here we say that expansion in the postprocessing gives rise to supercharged compression functions. Promising results were previously given by Lucks [22] in the iteration and Stam [37] for a

compression function. We develop a general theory and give two concrete examples based on the latter work.

4.1 Chopping: Compression in the Postprocessing

Let us consider an $m+s$-to-s bit compression function based on a single call to a blockcipher with key size k and block size n. In this section we will assume that $m+s=n+k$ and $s<n$. What can we say of the collision and preimage resistance of the compression function resp. iterated hash function, under which conditions will we achieve optimal security?

If we go through the criteria from the previous section, it is clear we can no longer satisfy them all. More to the point, whereas the first condition (bijectivity of the preprocessing) still applies, the postprocessing now becomes a mapping from n to s bits, which cannot be bijective since $s<n$. The natural generalization is to replace being a bijection with being balanced: all elements in the codomain should have the same number of preimages, namely 2^{n-s}. It turns out that this fairly simple modification works quite well. Again we have two types: the first one giving optimal collision and preimage resistance for the compression function; the second one giving optimal collision resistance in the iteration only (and guaranteed preimage resistance only up to the collision resistance).

Definition 13. *A single call blockcipher based compression function H^E is called chopped single call Type-I iff $s<n, m+k=n+s$, and the following three hold:*

1. *The preprocessing C^{PRE} is bijective.*
2. *For all M,V the postprocessing $C^{\mathrm{POST}}(M,V,\cdot)$ is balanced.*
3. *For all K,Y the modified postprocessing $C^{\mathrm{AUX}}(K,\cdot,Y)$ is balanced.*

Definition 14. *A single call blockcipher based compression function H^E is called chopped single call Type-II iff $s<n, m+k=n+s$ and the following three hold:*

1. *The preprocessing C^{PRE} is bijective.*
2. *For all M,V the postprocessing $C^{\mathrm{POST}}(M,V,\cdot)$ is balanced.*
3. *For all K the inverse preprocessing $C^{-\mathrm{PRE}}(K,\cdot)$ when restricted to its V output is balanced.*

Theorem 15. *Let H^E be a chopped single call Type-I compression function. Then the advantage of an adversary in finding a collision, resp. a preimage in H^E after q queries can be upper bounded by*

$$\mathsf{Adv}^{\mathsf{coll}}_H(q) \le q(q+1)/2^s, \qquad \mathsf{Adv}^{\mathsf{epre}}_H(q) \le q/2^{s-1}\ .$$

Proof. As before, a collision consists of two distinct pairs (M,V) and (M',V') satisfying either $H^E(M,V)=V_0$ or $H^E(M,V)=H^E(M',V')$. We will maintain a list of triples (M,V,W), initialized with $(-,-,V_0)$, such that $W=H^E(M,V)$ and the adversary has made the relevant queries to E and/or D. We need to bound the probability of a collision occurring in this list.

Bijectivity of C^{PRE} ensures that any query, be it forward or inverse, will add at most one triple (M,V,W) to this list of computable compression functions. As a result, after $i-1$ queries the list of computable values contains exactly i triples (M,V,W). We claim that the probability of the i'th query causing a collision with any of these triples is at most $2^{n-s}i/(2^n-(i-1))$.

Consider a forward query (K,X), which also fixes (M,V). Suppose that so far $t<i$ queries to E have been made involving key K. The answer to a fresh query to $E(K,\cdot)$ will therefore be distributed uniformly over a set of 2^n-t possible outcomes. Because C^{POST} is balanced for fixed (M,V), the number of preimages under C^{POST} of the i values W that are in the list of achieveable outputs so far is $2^{n-s}i$. Consequently, the probability of hitting one is at most $i2^{n-s}/(2^n-(i-1))$.

Similarly, consider an inverse query (K,Y). Then by balancedness of C^{AUX} there are $i2^{n-s}$ possibly outcomes of $D(K,Y)$ that would cause a collision. Again, assuming so far $t<i$ queries to E have been made involving key K, the answer to a fresh query to $D(K,\cdot)$ will be uniform over a set of 2^n-t. Hitting any of the forbidden outcomes has probability at most $i2^{n-s}/(2^n-i)$.

Using a union bound, the probability of a collision after q queries can then be upper bounded by $2^{n-s}\sum_{i=1}^{q} i/(2^n-i) \leq 2^{n-s-1}q(q+1)/(2^n-q)$. If $q \leq 2^{n-1}$ this can be upper bounded by $q(q+1)/2^s$ (and the bound is vacuous for larger q).

For preimages, the situation is similar. A straightforward adaptation of the arguments above will show that the probability of hitting the target image at the i'th query is at most $2^{n-s}/(2^n-(i-1))$. The probability of finding a preimage can then be upper bounded by union bound $2^{n-s}q/(2^n-q)$, which itself is upper bounded by $q/2^{s-1}$ for $q \leq 2^{n-1}$ (or vacuous otherwise). □

Theorem 16. *Let H^E be a chopped single call Type-II compression function. Then the advantage of an adversary in finding a collision in the iterated hash function $\mathcal{H}^H$ after q queries is upper bounded by*

$$\mathsf{Adv}^{\mathsf{coll}}_{\mathcal{H}}(q) \leq q(q+1)/2^s \ .$$

Proof. The proof is a fairly straightforward combination of that of Theorems 9 and 15.

Let $V_0 \in \{0,1\}^n$ be $\mathcal{H}$'s initial vector and consider the undirected graph $G = (V_G, E_G)$ as defined before, so to find a collision would require constructing a ρ-shape containing the initial vector V_0. Let T be the set of all nodes that are in a component containing a cycle or the initial vector V_0. Since we require C^{PRE} bijective, there are at most $i+1$ nodes in T after i queries. To complete a ρ-shape an edge has to be found of which both nodes are already part of T.

The probability that on the i'th query a collision is found by a forward query is at most $2^{n-s}i/(2^n-i)$: Y is uniformly distributed over a set of size at least 2^n-i and balancedness of C^{POST} means that each element in T has 2^{n-s} preimages Y under C^{POST} for any given (M,V). Therefore $C^{\mathrm{POST}}(M,V,Y)$ hits T at most with said probability.

Similarly, for an inverse query the probability of finding a collision on the i'th query using an inverse query is at most $2^{n-s}i/(2^n-i)$: this time balancedness of $C^{-\mathrm{PRE}}$ (when restricted to its V-part), implies that for any possible $V \in T$ there are exactly 2^{n-s} preimages X under $C^{-\mathrm{PRE}}$. Since X will be uniformly distributed over a set of size at least 2^n-i the claim follows.

We can now wrap up and conclude that the probability of finding a collision after q queries is at most $2^{n-s}\sum_{i=1}^{q} i/(2^n-i) \leq 2^{n-s-1}q(q+1)/(2^n-q)$. If $q \leq 2^{n-1}$ this can be upper bounded by $q(q+1)/2^s$ (and the bound is vacuous for larger q). □

4.2 Overloading: Compression in the Preprocessing

Another way to improve efficiency it to keep $s = n$, but allow $m > k$. In this case bijectivity of the preprocessing can no longer be satisfied, which has ramifications throughout.

Firstly, for a given pair (K,X) it is now the case that $C^{-\mathrm{PRE}}$ yields a set of 2^{m-k} pairs (M,V). Consequently, the modified postprocessing $C^{\mathrm{AUX}}(K,\cdot,Y)$ becomes a function from n-bits to subsets of size (up to) 2^{m-k} of $\{0,1\}^n$. Our requirement on this new type of C^{AUX} is a natural generalization of balancedness.

Secondly, although the condition that $C^{\mathrm{POST}}(M,V,\cdot)$ is bijective is still well-defined, it is no longer sufficient. For instance, if $C^{\mathrm{PRE}}(M,V) = C^{\mathrm{PRE}}(M',V')$ for certain values of $(M,V) \neq (M',V')$ and the bijections $C^{\mathrm{POST}}(M,V,\cdot)$ and $C^{\mathrm{POST}}(M',V',\cdot)$ are identical, then collisions can very easily be found. To avoid this problem we explicitly rule out collisions in the output whenever (M,V) and (M',V') already collide during preprocessing (in C^{PRE}).

Definition 17. *A single call blockcipher based compression function H^E is called overloaded single call Type-I iff $s = n, m \geq k$, and the following four hold:*

1. *The preprocessing C^{PRE} is balanced.*
2. *For all $(M,V) \neq (M',V')$ with $C^{\mathrm{PRE}}(M,V) = C^{\mathrm{PRE}}(M',V')$ and all Y it holds that $C^{\mathrm{POST}}(M,V,Y) \neq C^{\mathrm{POST}}(M',V',Y)$.*
3. *For all M,V the postprocessing $C^{\mathrm{POST}}(M,V,\cdot)$ is bijective.*
4. *For all K,Y the modified postprocessing $C^{\mathrm{AUX}}(K,\cdot,Y)$ is balanced in the sense that for all V the number of X such that $V \in C^{\mathrm{AUX}}(K,X,Y)$ equals 2^{m-k}.*

Theorem 18. *Let H^E be an overloaded single call Type-I compression function. Then the advantage of an adversary in finding a collision, resp. a preimage in H^E after q queries can be upper bounded by*

$$\mathsf{Adv}^{\mathsf{coll}}_H(q) \le q(q+1)/2^{2k+n-2m}, \qquad \mathsf{Adv}^{\mathsf{epre}}_H(q) \le q/2^{n+k-m-1} .$$

Proof. As before, a collision consists of two distinct pairs (M, V) and (M', V') satisfying either $H^E(M, V) = V_0$ or $H^E(M, V) = H^E(M', V')$ We will maintain a list of triples (M, V, W), initialized with $(-, -, V_0)$, such that $W = H^E(M, V)$ and the adversary has made the relevant queries to E and/or D. We need to bound the probability of a collision occurring in this list.

Balancedness of C^{PRE} ensures that any query, be it forward or inverse, will add exactly 2^{m-k} triples (M, V, W) to this list of computable compression functions. As a result, after $i-1$ queries the list of computable values contains exactly $(i-1)2^{m-k}+1$ triples (M, V, W). We claim that the probability of the i'th query causing a collision with any of these triples is at most $i2^{2(m-k)}/(2^n - i)$.

Consider a forward query (K, X). There are 2^{m-k} corresponding pairs (M, V), due to C^{PRE} being balanced. Because C^{POST} is bijective for any fixed (M, V), for each tuple $((M, V), W)$ there is exactly one possible outcome of the blockcipher that would cause a collision. Here we need to range over all (M, V) corresponding to the query (K, X) at hand, and all W in the list of achieveable outputs so far. Combined that gives $2^{m-k}((i-1)2^{m-k}+1) < i2^{2(m-k)}$ tuples. Now suppose that so far $t < i$ queries to E have been made involving key K. The answer to a fresh query to $E(K, \cdot)$ will therefore be distributed uniformly over a set of $2^n - t$ possible outcomes. Hence the probability of a collision based on this query is at most $i2^{2(m-K)}/(2^n - i)$.

Similarly, consider an inverse query (K, Y). Then by balancedness of C^{AUX}, for each of the $(i-1)2^{m-k}+1 < i2^{m-k}$ triples (M, V, W) there are 2^{m-k} possible outcomes of $D(K, Y)$ that would cause a collision. Again, assuming so far $t \le i$ queries to E have been made involving key K, the answer to a fresh query to $D(K, \cdot)$ will be uniform over a set of $2^n - t$. Hitting any of the forbidden outcomes has probability at most $i2^{2(m-k)}/(2^n - i)$.

Using a union bound, the probability of a collision after q queries can then be upper bounded by $2^{2(m-k)} \sum_{i=1}^{q} i/(2^n - i) \le 2^{2(m-k)-1} q(q+1)/(2^n - q)$. If $q \le 2^{n-1}$ this can be upper bounded by $q(q+1)/2^{2k+n-2m}$ (and the bound is vacuous for larger q).

For preimages, the situation is similar. A straightforward adaptation of the arguments above will show that the probability of hitting the target image at the i'th query is at most $2^{m-k}/(2^n - i)$. The probability of finding a preimage can then be upper bounded by union bound $2^{m-k}q/(2^n - q)$, which itself is upper bounded by $q/2^{n+k-m-1}$ for $q \le 2^{n-1}$ (or vacuous otherwise). □

Theorem 18 can be reinterpreted by saying that to find collisions roughly $2^{n/2+k-m}$ queries are required; to find preimages roughly 2^{n+k-m} queries should suffice. It is interesting to compare the collision resistance thus achieved with recently conjectured optimal bounds [34, 37]. A straightforward generalization of Rogaway and Steinberger's result [34] suggests the best we can achieve is collision resistance up to $2^{n/2+k-m}$ queries, neatly corresponding to our construction. However, Stam [37] conjectures collision resistance is feasible up to $2^{(n+k-m)/2}$ queries, based on an ideal state size s of $n+k-m$ bits. Using this state size actually brings us back exactly to compression in the postprocessing as discussed in the previous section: by reducing s we can increase m while maintaining $n+k = m+s$ and Theorem 15 essentially guarantees collision resistance up to $2^{(n+k-m)/2}$ queries. So here is another scenario where reducing the state size mysteriously seems to boost collision resistance.

But all is not as it seems. An example overloaded single call Type-I compression function is Davies-Meyer with the $m-k$ superfluous message bits xored directly into the output. It is not hard to show that in this case the collision finding advantage is much smaller than Theorem 18 makes believe:

$$\mathsf{Adv}^{\mathsf{coll}}_H(q) \le q(q+1)/2^{k+n-m} .$$

Iterated Case. For rate-1 and chopped compression functions, looking at the iteration gave rise to a second class of schemes that had the same collision resistance in the iteration as the main schemes,

but inferior preimage resistance. For overloaded compression functions, we do not give a classificiation of Type-II schemes (also in light of our Type-I bounds' lack of tightness). However, we do point out that some non-trivial results in this setting were previously achieved for sponge functions [6], whose collision resistance (in the iteration) holds roughly up to $2^{(n-m)/2}$ queries ($k = 0$). This matches the collision resistant compression function of the previous paragraph.

However, recent developments indicate that iteration might boost collision resistance even further. In particular, the sponge construction has rate $\alpha = m/(n-m)$ achieving collision resistance up to roughly $2^{n(1-\alpha)/2}$ queries. Rogaway and Steinberger [34] have shown that for any rate-α construction after $1.9n2^{n(1-\alpha)}$ queries collisions are guaranteed. This still leaves a considerable gap.

4.3 Supercharging: Expansion in the Postprocessing

Whereas for chopped and overloaded compression functions we sacrificed security for the sake of efficiency, in this section we will attempt the exact opposite: sacrificing efficiency for the sake of security. We do this by extending the state size, so $s > n$. Not to complicate things further, we will assume that $m + s = n + k$ (and let C^{PRE} be bijective). For any fixed pair (M, V) we have that C^{POST} maps $\{0,1\}^n$ to $\{0,1\}^s$. Since $n < s$ this cannot be a bijection, but at best an injection (similar for C^{AUX}). If all these injections have exactly the same range, we are not using our codomain of 2^s values to the full; indeed we might have well been padding the state with a constant. This leads us to the following formalization.

Definition 19. *A single call blockcipher based compression function H^E is called supercharged single call Type-I with overlap γ iff $s \geq n, m + s = n + k$ and the following three hold:*

1. *The preprocessing C^{PRE} is bijective.*
2. *For all M, V the postprocessing $C^{\text{POST}}(M, V, \cdot)$ is injective, with effective range $R_{\text{POST},(M,V)}$.*
3. *For all K, Y the modified postprocessing $C^{\text{AUX}}(K, \cdot, Y)$ is injective, with effective range $R_{\text{AUX},(K,Y)}$.*

Where the overlap γ is defined as:

$$\gamma = \max\left\{|R_Z \cap R_{Z'}| : Z, Z' \in \{\text{POST}, \text{AUX}\} \times \{0,1\}^{k+n}, Z \neq Z'\right\} .$$

Theorem 20. *Let H^E be a supercharged single call Type-I compression function with overlap γ. Then the advantage of an adversary in finding a collision after $q \leq 2^{n-1}$ queries can be upper bounded by*

$$\mathsf{Adv}^{\mathsf{coll}}_H(q) \leq q\kappa/2^{n-1} + 2^{m+s+1}\left(\frac{e\gamma q}{(\kappa-1)2^{n-1}}\right)^{\kappa-1}$$

for arbitrary positive integer $\kappa > q\gamma/2^{n-1}$.

Proof. As usual, we want to upper bound the probability of finding a collision on the i'th query ($i \leq q$), so we can sum over all i up to q for a union bound of the total success probability. Because C^{PRE} is bijective, we know that each query adds exactly one triple (M, V, W) to the set of computable digests. Thus after $i-1$ queries, and assuming no collisions have been found so far, we will have a set of i computable digests (including the initial vector V_0). Let's call this set $T(i-1)$.

For the i'th query to give us a collision it should result in a digest W in the set $T(i-1)$. But we know that the digest will certainly be in R_Z with $Z = (\text{POST}, K, X)$ or $Z = (\text{AUX}, K, Y)$ corresponding to the query made. Moreover, the digest will be uniformly distributed over R_Z, up to the usual correction taking care of previous queries to E or D using the same key. Hence the probability of a collision on the i'th query can be upper bounded by $|R_Z \cap T(i-1)|/(2^n - i)$. For any given positive integer κ, let p_κ be the probability that $\max_Z |R_Z \cap T(q)| > \kappa$. Then the total probability of finding a collision can be upper bounded by $p_\kappa + \sum_{i=1}^{q} \kappa/(2^n - q)$, where as usual we can upper bound the second term by $q\kappa/2^{n-1}$ (or get a vacuous bound).

What remains is bounding p_κ. Fix Z, then the probability that the i'th query $Z' \neq Z$ gives an answer in R_Z is upper bounded by $\gamma/(2^n - i) < \gamma/2^{n-1}$. We will assume that $V_0 \in R_Z$ and upper bound

the probability that $|R_Z \cap T(q)|$ exceeds κ by a binomial distribution with q repetitions and probability $\gamma/2^{n-1}$. For $\kappa > q\gamma/2^{n-1}$ this is allowed. A Chernoff bound followed by a union bound then allows us to bound by:

$$\begin{aligned}\mathsf{Pr}\left[\max_Z |R_Z \cap T(q)| \geq \kappa\right] &\leq \sum_Z \mathsf{Pr}\left[|R_Z \cap T(q)| \geq \kappa\right] \\ &\leq 2^{m+s+1}\mathsf{Pr}\left[B[q;\gamma/2^{n-1}] \geq \kappa - 1\right] \\ &\leq 2^{m+s+1}\left(\frac{e\gamma q}{(\kappa-1)2^n}\right)^{\kappa-1} .\end{aligned}$$

Collecting the two terms gives the stated upper bound. □

Corollary 21. *Let H^E be a supercharged single call Type-I compression function with overlap γ. Then for $q < 2^{n-1}/\gamma^{\frac{1}{2}}$ the probability of finding a collision can be upper bounded by*

$$\mathsf{Adv}^{\mathsf{coll}}_H(q) \leq 2\max(2e\gamma^{\frac{1}{2}}, m+n+s+2)q/2^n .$$

Proof. First we will voluntarily strengthen the restriction on κ in Theorem 20 to $\kappa > q\gamma e/2^{n-2}$. Since we only allow $q < 2^{n-1}/\gamma^{\frac{1}{2}}$ this is satisfied for all $\kappa > 2e\gamma^{1/2}$. This has the effect that we can upper bound the second term of the bound on the advantage by $2^{m+s+2-\kappa}$. By additionally requiring that $\kappa > n+m+s+2$ we ensure that the whole second term is upper bounded by 2^{-n}, making the first term dominant. For simplicity, we simply doubled it in the bound stated in the corollary. □

In practice this means that we get good security up to q of order $2^n/\gamma^{\frac{1}{2}}$. Stam [37] suggests that finding collisions can be expected after $2^{(n+k-m)/2}$ queries. Since $n+k = m+s$ this neatly corresponds to $2^{s/2}$, in other words optimal collision resistant compression functions of this type might actually exist. Note that the rate is lower than before, arguably m/n. As we show in Lemma 22, the best we can hope for is γ of order 2^{2n-s}, giving collision resistance up to $2^{s/2}$ queries. Whether for all relevant settings of n, s, k, and m there exists a postprocessing C^{POST} with overlap γ close to 2^{2n-s} is an open problem. Below we give two examples where it does though, based on an earlier construction [37].

Lemma 22. *Let H^E be a supercharged single call Type-I compression function then overlap*

$$\gamma \geq \frac{2(2^{2n+m}-2^n)}{2^{s+m}-1} \quad (\approx 2^{2n-s+1}) .$$

Proof. If for some $Z \neq Z' \in \{\text{POST}, \text{AUX}\} \times \{0,1\}^{n+k}$ it holds that $R_Z = R_{Z'}$ we have that $\gamma = |R_Z| = 2^{n+k}$ by injectivity of C^{POST} (and $n+k > n > 2n-s$). Henceforth assume that all the R_Z are distinct. In that case $\left\{R_Z | Z \in \{\text{POST}, \text{AUX}\} \times \{0,1\}^{n+k}\right\}$ can be regarded as a constant weight code of length 2^s, weight 2^n and cardinality 2^{s+m+1}. The minimum distance d of the code can be related to the overlap γ of the compression function by $d = 2 \cdot 2^n - \gamma$. Standard bounds on the maximum cardinality of a constant weight code of given length, weight and minimum distance gives us the stated bound on γ.

In particular, substituting the various parameters in a relaxed Johnson bound [10, VII.1.12, 1.80.2] yields

$$2^{s+m} \leq \frac{2^s(2^n-\gamma/2)}{2^{2n}-2^{n+s+m}+2^{s+m}(2^n-\gamma/2)}$$

which simplifies to

$$2^m \leq \frac{2^n-\gamma/2}{2^{2n}-2^{m+s}\gamma/2}$$

which solves for γ as

$$\gamma \geq \frac{2(2^{2n+m}-2^n)}{2^{s+m}-1} \quad (\approx 2^{2n-s+1}) .$$

□

We have shown that supercharged single call Type-I compression functions potentially provide a boost of collision resistance beyond the birthday bound on the output length of the underlying block-cipher. A natural question is whether the increased state size also provides us with better preimage resistance. The answer to this question unfortunately is no. Essentially preimage resistance for supercharged single call Type-I compression functions depends only on the block length n and not on the state size s (nor on the overlap γ of the construction).

Theorem 23. *Let H^E be a supercharged single call Type-I compression function. Then the advantage of an adversary in finding a preimage after q queries can be upper bounded by*

$$\mathsf{Adv}^{\mathsf{epre}}_H(q) \leq q/2^{n-1} \; .$$

Proof. Let target digest W be given. We want to upper bound the probability of finding a preimage on the i'th query ($i \leq q$), so we can sum over all i up to q for a union bound of the total success probability. Because C^{PRE} is bijective, we know that after each query the adversary is (at best) able to compute the compression function for a single new input. Since there are at least $2^n - (i-1)$ equally likely possible responses to the query, and each response (by injectivity of C^{POST} and C^{AUX}) corresponds to a distinct possible digest, the probability of hitting the target digest W on the i'th query is upper bounded by $1/(2^n - i + 1)$. Hence the total probability of finding a preimage can be upper bounded by $\sum_{i=1}^{q} 1/(2^n - i + 1) \leq q/(2^{n-q})$, which as usual we can upper bound by $q/2^{n-1}$ (or become vacuous).

At the same time, an adversary exists that comes close to achieving this bound, where we make the assumption that the target image does have a preimage (under an instantiation of E consistent with the oracle for E as queried by the adversary). In fact, the attack we demonstrate works for *any* supercharged single call compression function. We use the notation spre' to signal that determining a target digest is not in the range also counts as a success (cf. [2]).

Proposition 24. *Let H^E be a supercharged single call compression function. Then there exists an adversary whose advantage in finding a preimage (or determining no preimage exists) after q queries can be lower bounded by*

$$\mathsf{Adv}^{\mathsf{spre}'}_H(q) \geq 1 - (1 - 2^{-n})^q \; .$$

Proof. Let $W \in \{0,1\}^s$ be the target digest. Consider the following adversary. It starts by determining the set of (M, V) for which $W \in R_{\mathrm{POST},(M,V)}$. For any given (M, V) in this set, it computes the corresponding $(K, X) \leftarrow C^{\mathrm{PRE}}(M, V)$. If no queries have been asked yet, querying $E(K, X)$ on one of these (K, X)-pairs has a probability of at least 2^{-n} of hitting the right value in the corresponding $R_{\mathrm{POST},(M,V)}$; hence yielding a preimage of W. After a query is answered (without yielding a preimage), the adversary updates its list (K, X) that could hit the digest (in particular, if the current query is (K, X'), its answer could rule out that a later query (K, X) hits the digest). As long as prior queries have not yet yielded a preimage, we can maintain a success probability of at least 2^{-n} per query. (Since we have successfully determined no preimage exists if we run out of useful queries (K, X) to ask.) The lower bound on the advantage follows from upper bounding the failure probability.

It follows that the expected number of queries needed to find a preimage is upper bounded by 2^n. This upper bound on the preimage resistance is significantly tighter than the general bounds by Stam [37, Theorem 8] (cf. Rogaway and Steinberger [34, Theorem 2]) for the parameters in question (indeed, those bounds are essentially vacuous here, stating that preimages can be found after 2^{2n} queries).

The attack we described finds preimages in the compression function. One can wonder to what extent preimages in the related iterated hash function can be found. Using a well-known meet-in-the-middle attack [19] this would imply preimage for the iterated hash functions can be found using (a small constant multiple of) $2^{(n+s)/2}$ queries. The only catch with this attack is that we need to find $2^{(s-n)/2}$ preimages (building a tree rooted at the target digest). It is not necessarily true that all chaining variables involved always have a preimage (moreover, determining for a specific chaining variable that has no

preimage does not yield a success, it simply connotes a dead end). In other words, we might get stuck unless we make a suitable (and seemingly reasonable) uniformity assumption. In that case the attack is better than the ideal preimage resistance of 2^s queries of course, provided $n < s$. Note that if we are in fact promised that for all $V, W \in \{0,1\}^s$ there exists an $M \in \{0,1\}^n$ with $W \in R_{\text{POST},(M,V)}$ the generic attack does work directly on the iterated hash function (with a preimage found after an expected 2^{n+1} queries). We will see an example of this later. (We also note that using a generalized Merkle-Damgård transform with a sufficiently preimage resistant postprocessing could thwart this attack, however for second-preimage resistance postprocessing seems inherently ineffective in boosting security.)

Example I: A Double-Length Construction. We recall the construction [37] for a double length compression function based on a single ideal $3n$-to-n compression function F. Split the $2n$-bit state V in two equally sized parts V_1 and V_2. Then given an n-bit message block M, compression proceeds as follows:

1. Compute $Y \leftarrow F(M, V_1, V_2)$.
2. Output $(W_1, W_2) \leftarrow (Y, V_2Y^2 + V_1Y + M)$.

where the polynomial evaluation is over $\mathbb{F}_{2^n}$. Originally only a proof of collision resistance against non-adaptive adversaries was given, based on random functions instead of random permutations (so in particular an adversary would not have access to an inversion oracle). We would like to port the scheme to the ideal cipher model, based on a blockcipher with $k = 2n$.

1. Set $K \leftarrow (V_1, V_2)$ and $X \leftarrow M$.
2. Compute $Y \leftarrow E(K, X)$.
3. Compute $W_1 \leftarrow Y + M$ and $W_2 \leftarrow MW_1^2 + V_1W_1 + V_2$; output (W_1, W_2).

Lemma 25. *For the compression function above, $\gamma = 3$.*

Proof. To determine the overlap γ it helps to first write down the effective ranges $R_{\text{POST},(M,V)}$ and $R_{\text{AUX},(K,Y)}$ explicitly. It is easy to see that

$$R_{\text{POST},(M,V_1,V_2)} = \left\{(W, MW^2 + V_1W + V_2) | W \in \{0,1\}^n\right\}$$

and with a little bit more effort, using that $M = Y + W$ and $(K_1, K_2) = (V_1, V_2)$,

$$R_{\text{AUX},(K_1,K_2,Y)} = \left\{(W, W^3 + YW^2 + K_1W + K_2) | W \in \{0,1\}^n\right\} .$$

As a result, for (W_1, W_2) to be in the intersection of R_Z and $R_{Z'}$, we require W_1 to be a root of the difference of the two polynomials that define W_2 for Z resp. Z'. It can be readily verified that $Z \neq Z'$ implies the relevant two polynomials are distinct as well, and the resulting difference is a non-zero polynomial of degree at most three. It will therefore have at most three roots over $\mathbb{F}_{2^n}$. □

Corollary 26. *For the compression function above, for $q \leq 2^{n-\frac{3}{2}}$:*

$$\mathsf{Adv}^{\mathsf{coll}}_H(q) \leq (n + \frac{1}{2})q/2^{n-3} .$$

Curiously, if we would change the computation of W_2 even slightly, for instance $W_2 \leftarrow V_2W_1^2 + V_1W_1 + M$, the impact on the overlap γ is dramatic. Suddenly $R_{\text{AUX},(K_1,K_2,Y)} = \{W, K_2W^2 + (K_1 + 1)W + Y | W \in \{0,1\}^n\}$ and consequently $R_{\text{AUX},(V_1+1,V_2,M)} = R_{\text{POST},(V_1,V_2,M)}$, so that $\gamma = 2^n$. As a result, Theorem 20 can only be used to guarantee collision resistance up to roughly $2^{n/2}$ queries.

Nonetheless, with a more refined analysis we can still prove collision resistance close to the birthday bound. The main point here is that for each $Z \in \{\text{POST}, \text{AUX}\} \times \{0,1\}^{k+n}$ there exists only one (a unique) distinct Z^* achieving the high overlap. For all other Z' the intersection is small, indeed $|R_Z \cap R_{Z'}| \leq 2$. As a result, if we look at the proof of Theorem 20 we can still bound $|R_Z \cap T(i-1)|$, almost exactly the same as before: we assume that Z^* was one of the previous queries (which will add exactly one to the intersection at hand), for all remaining queries we can use the modified overlap $\gamma' \leq 2$.

Example II: An Intermediate Construction. We conclude with a construction based on a $3n/2$ bit state (split into three parts of $n/2$ bits each), that compresses $n/2$ message bits.

1. $X \leftarrow (M, V_1), K \leftarrow (V_2, V_3)$;
2. $Y \leftarrow E(K, X)$;
3. $W_1 \leftarrow Y_1 + M, W_2 \leftarrow Y_2 + V_1$, and $W_3 \leftarrow MW_1^3 + V_1W_1^2 + V_2W_1 + V_3$.

Lemma 27. *For the compression function above,* $\gamma = 2^{2+n/2}$.

Proof. The proof is similar to that of Lemma 25, but this time W_1 needs to be a root of a fourth degree polynomial and W_2 is completely free. The key equation is

$$R_{\text{AUX},(K_1,K_2,Y_1,Y_2)} = \left\{(W_1, W_2, W_1^4 + Y_1W_1^3 + (Y_2 + W_2)W_1^2 + K_1W_1 + K_2) | W \in \{0,1\}^n\right\} .$$

□

Corollary 28. *For the compression function above and all* $q < 2^{3n/4-2}$

$$\mathsf{Adv}_H^{\mathsf{coll}}(q) \leq eq/2^{3n/4-3} .$$

Preimage Resistance and Other Properties Stam [37] presented his double length construction mainly as a proof-of-concept, claiming obvious shortcomings without being specific. Similarly, the two supercharged compression functions (and the corresponding MD-iterated hashes) have issues. We will elaborate for the double length construction.

Because of the way V_2 is constructed, it is easy to distinguish the compression function from a true random oracle after just one call (to the compression function). If the compression function is MD-iterated and only a single message block is being hashed, the resulting output allows easy recovery of this message (since the incoming state is the known initial vector). As also noted by Bagheri et al. [3] (as well as by an FSE'09 referee) the double length construction allows a preimage attack with expected query-complexity of around $q \approx 2^{n+1}$. We have given the general attack discussing Proposition 24, here it is specific to our double length construction.

Given any target chaining variable (W_1, W_2), generate a fresh input chaining variable (V_1, V_2) (by picking a random first message block in the iterated hash) and set $M = (W_2 + V_1W_1 + V_2)/W_1^2$. This choice of M ensures that if a partial preimage for W_1 is found, it immediately yields a full preimage. That is, if $E((V_1, V_2), M) = W_1 + M$ then a preimage is found. Since E is random, the probability of success after a single try is 2^{-n}; so an expected number of 2^n tries (each costing two blockcipher calls: creating a fresh input chaining variable and checking for a preimage) suffices.

For completeness, Bagheri et al. [3] also mention a near-collision attack on the original construction from Stam [37], that straightforwardly applies to the current blockcipher based version as well. More precisely, they show how to find near-collisions in the compression function in $3n/2$ out of the $2n$ bits in time about $2^{n/2}$ (the attack however does not seem to generalize to the iterated hash function).

Although the shortcomings described above limit the attractivity of our double length proposal for the purpose of construction a hash function with larger output length, it might still be useful in construction a single-length proposal with a wide-pipe [21]. In particular, if we have two independent blockciphers E_1 and E_2, we can create a hash function by first computing (W_1, W_2) using our MD-iterated double length hash function based on E_1 and outputting $E_2(W_1, W_2, 0^n)$. We claim that this hash function is indifferentiable up to almost $2^n/n$ queries by using the preimage awareness framework. Since $E_2(W_1, W_2, 0^n)$ can easily be seen to be a random oracle from $2n$-bits to n-bits (and independent from E_1 by assumption), all we need to show is the adaptive preimage resistance of the compression function up to (at least) $2^n/n$ queries. However, this can be shown using essentially the same proof as used for standard Davies-Meyer.

We note that known rate-1/2 double length constructions with optimal collision resistance, such as Hirose's, tandem-DM or Lucks' slight variation thereof, are likely to be adaptively preimage resistance

as well. Moreover, these schemes are less brittle, in the sense that no $O(2^n)$ preimage attacks are known against them. From a practical point of view, the rate-1/2 schemes might even be more efficient than our rate-1 scheme for the simple reason that the two finite field multiplications we need (per compression function call) are likely to be more expensive than the extra blockcipher call for the rate-1/2 schemes. Nonetheless, we have shown that the second blockcipher call is not strictly necessary: designing a rate-1 double length secure scheme with less computational overhead remains an open problem.

5 Conclusion

We have presented a general framework to capture blockcipher based hash functions, in particular those that iterate a compression function based on a single call only. This has allowed us to develop a deeper understanding of existing result concerning the security of PGV schemes. We also extended the framework to several scenarios where input and output of the compression function do not match those of the underlying blockcipher.

For chopped compression functions our framework allows for a straightforward extension. For overloaded compression functions we do not get tight security bounds; constructions are known that best them. Security in the iteration is pointed out as an open problem. Finally, for collision resistance beyond the cipher's blocksize we develop a general theory and construct two schemes accordingly.

Acknowledgements The author would like to thank John Black, Phil Rogaway and Onur Özen for useful ideas on the presentation; Elena Andreeva, Lars Knudsen and the anonymous FSE'09 referees for pointing out certain problems with preimage resistance; and Tom Shrimpton for great advice for the duration of the project.

References

[1] E. Andreeva, G. Neven, B. Preneel, and T. Shrimpton. Seven-property-preserving iterated hashing: Rox. In K. Kurosawa, editor, *Advances in Cryptography—Asiacrypt'07*, volume 4833 of *Lecture Notes in Computer Science*, pages 130–146. Springer-Verlag, 2007.

[2] E. Andreeva and M. Stam. Another glance at preimage resistance, 2009. Manuscript.

[3] N. Bagheri, L. R. Knudsen, M. Naderi, and S. S. Thomsen. Hash functions and information theoretic security, 2009. Manuscript.

[4] D. J. Bernstein. Cubehash specification (2.b.1). Submission to NIST, 2008.

[5] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. Sponge functions. Ecrypt Hash Workshop, 2007.

[6] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. On the indifferentiability of the sponge construction. In Smart [36], pages 181–197.

[7] J. Black. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In M. J. Robshaw, editor, *FSE'06*, volume 4047 of *Lecture Notes in Computer Science*, pages 328–340. Springer-Verlag, 2006.

[8] J. Black, P. Rogaway, and T. Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In M. Yung, editor, *Advances in Cryptography—Crypto'02*, volume 2442 of *Lecture Notes in Computer Science*, pages 320–335. Springer-Verlag, 2002.

[9] G. Brassard, editor. *Advances in Cryptography—Crypto'89*, volume 435 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[10] C. J. Colbourn and J. H. Dinitz. *Handbook of Combinatorial Designs*. CRC, 2007.

[11] J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård revisited: How to construct a hash function. In V. Shoup, editor, *Advances in Cryptography—Crypto'05*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer-Verlag, 2005.

[12] I. Damgård. A design principle for hash functions. In Brassard [9], pages 416–427.

[13] Y. Dodis, T. Ristenpart, and T. Shrimpton. Salvaging Merkle-Damgård for practical applications. In A. Joux, editor, *Advances in Cryptography—Eurocrypt'09*, volume 5479 of *Lecture Notes in Computer Science*, page to appear. Springer-Verlag, 2009.

[14] L. Duo and C. Li. Improved collision and preimage resistance bounds on PGV schemes. Technical Report 462, IACR's ePrint Archive, 2006.

[15] P. Flajolet and A. M. Odlyzko. Random mapping statistics. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptography—Eurocrypt'89*, volume 434 of *Lecture Notes in Computer Science*, pages 329–354. Springer-Verlag, 1990.

[16] S. Hirose. Provably secure double-block-length hash functions in a black-box model. In C. Park and S. Chee, editors, *ICISC'04*, volume 3506 of *Lecture Notes in Computer Science*, pages 330–342. Springer-Verlag, 2005.

[17] W. Hohl, X. Lai, T. Meier, and C. Waldvogel. Security of iterated hash functions based on block ciphers. In Stinson [38], pages 379–390.
[18] L. R. Knudsen, C. Rechberger, and S. S. Thomsen. The Grindahl hash functions. In A. Biryukov, editor, *FSE'07*, volume 4593 of *Lecture Notes in Computer Science*, pages 39–57. Springer-Verlag, 2007.
[19] X. Lai and J. L. Massey. Hash function based on block ciphers. In R. A. Rueppel, editor, *Advances in Cryptography—Eurocrypt'92*, volume 658 of *Lecture Notes in Computer Science*, pages 55–70. Springer-Verlag, 1992.
[20] J. Lee and J. H. Park. Adaptive preimage resistance and permutation-based hash functions. Technical Report 66, IACR's ePrint Archive, 2009.
[21] S. Lucks. A failure-friendly design principle for hash functions. In B. K. Roy, editor, *Advances in Cryptography—Asiacrypt'05*, volume 3788 of *Lecture Notes in Computer Science*, pages 474–494. Springer-Verlag, 2005.
[22] S. Lucks. A collision-resistant rate-1 double-block-length hash function. In E. Biham, H. Handschuh, S. Lucks, and V. Rijmen, editors, *Symmetric Cryptography*, number 07021 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2007. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.
[23] S. Matyas, C. Meyer, and J. Oseas. Generating strong one-way functions with cryptographic algorithms. *IBM Technical Disclosure Bulletin*, 27(10a):5658–5659, 1985.
[24] U. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In M. Naor, editor, *TCC'04*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer-Verlag, 2004.
[25] A. Menezes, P. van Oorschot, and S. Vanstone. *CRC-Handbook of Applied Cryptography*. CRC Press, 1996.
[26] R. C. Merkle. One way hash functions and DES. In Brassard [9], pages 428–446.
[27] S. Miyaguchi, M. Iwata, and K. Ohta. New 128-bit hash function. In *Proceedings 4th International Joint Workshop on Computer Communications*, pages 279–288, 1989.
[28] O. Özen and M. Stam. Double length blockcipher based hashing, 2008. Manuscript.
[29] J. Pollard. Kangaroos, monopoly and discrete logarithms. *Journal of Cryptology*, 13(4):437–447, 2000.
[30] N. Pramstaller, M. Lamberger, and V. Rijmen. Second preimages for iterated hash functions and their implications on macs. In J. Pieprzyk, H. Ghodosi, and E. Dawson, editors, *ACISP'07*, volume 4586 of *Lecture Notes in Computer Science*, pages 68–81. Springer-Verlag, 2007.
[31] B. Preneel, R. Govaerts, and J. Vandewalle. Hash functions based on block ciphers: A synthetic approach. In Stinson [38], pages 368–378.
[32] R. L. Rivest. The md6 hash function – a proposal to nist for sha-3. Submission to NIST, 2008.
[33] P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications and separations for preimage resistance, second-preimage resistance, and collision resistance. In B. K. Roy and W. Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer-Verlag, 2004.
[34] P. Rogaway and J. Steinberger. Security/efficiency tradeoffs for permutation-based hashing. In Smart [36], pages 220–236.
[35] A. Ross, T. Shrimpton, and M. Stam. A new permutation-based variable-input-length variable-output-length hash function, 2009. Manuscript.
[36] N. P. Smart, editor. *Advances in Cryptography—Eurocrypt'08*, volume 4965 of *Lecture Notes in Computer Science*. Springer-Verlag, 2008.
[37] M. Stam. Beyond uniformity: Better security/efficiency tradeoffs for compression functions. In D. Wagner, editor, *Advances in Cryptography—Crypto'08*, volume 5157 of *Lecture Notes in Computer Science*, pages 397–412. Springer-Verlag, 2008.
[38] D. Stinson, editor. *Advances in Cryptography—Crypto'93*, volume 773 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.
[39] H. Wu. The hash function jh. Submission to NIST, 2008.

A Unreal Collisions

According to our and Black et al.'s results [8], the (Type-II) rate-1 blockcipher based compression function $H^E(M,V) = E(M,V)$ is collision-resistant in the iteration, indeed it was proven that (cf. Corollary 12) $\mathsf{Adv}^{\mathsf{coll}}_{\mathcal{H}}(q) \leq \frac{1}{2}q(q+1)/(2^n - q)$. In this appendix we show that in fact it is possible to find collisions in the hash function (with probability 1) without making any queries to the underlying blockcipher. Although in this case the adversary does not know to which value his messages collide, a single query suffices to amend this problem. This paradox can be explained by the assumption we used to prove the bound from Corollary 12 (and all the other bounds in the main body of this paper), namely that the adversary *honestly* evaluates the hash function on the messages it outputs. Since the colliding messages are very long, honest evaluation would consume a large number of queries and hence require a sufficiently large q to make our collision bounds vacuous. (Another way out of the paradox would be to use time complexity instead of query complexity and remark that time complexity is at least query complexity plus output size.)

Proposition 29. *Let $n, k > 0$ be integer parameters. Let $H : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a compression function taking oracle $E \in \mathrm{Block}(k,n)$ defined by $H^E(M,V) = E(M,V)$ and let $\mathcal{H}$ be its iterated hash function. Define $L = k(2^n!)$. Then $\mathbf{M} = 1^k 0^L$ and $\mathbf{M}' = 0^L 1^k$ collide under $\mathcal{H}$.*

Proof. Define $\mathcal{H}_V(\mathbf{M})$ for $V \in \{0,1\}^n$ and $\mathbf{M} \in (\{0,1\}^k)^*$ as (vanilla) MD-iterated hash with initial vector V (based on compression function H as defined above). Our claim is that for all $V \in \{0,1\}^n$ it holds that $\mathcal{H}_V(0^L) = V$ and consequently that $\mathcal{H}(1^k 0^L) = \mathcal{H}(1^k) = \mathcal{H}(0^L 1^k)$ proving the proposition.

For the claim, first observe that the string 0^L will be parsed by $\mathcal{H}$ as $2^n!$ blocks of k bits each. Moreover, all blocks are equal, namely 0^k. Writing π for the permutation $E(0,\cdot)$, we get that processing one block leads to $\mathcal{H}_V(0^k) = E(0^k, V) = \pi(V)$ and multiple blocks to $\mathcal{H}_V((0^k)^\ell) = \pi^\ell(V)$. In particular, $\mathcal{H}_V(0^L) = \pi^{(2^n!)}(V)$. Regarding π as an element in the group of all permutations on 2^n elements, we can conclude that $\pi^{(2^n)!}$ equals the identity permutation, since $(2^n)!$ is a multiple of the exponent of the group of permutations (indeed, one could also use the smaller $\mathrm{lcm}(1,\ldots,2^n)$ instead). Hence $\mathcal{H}_V(0^L) = \pi^{(2^n!)}(V) = V$ as claimed.

A.1 Variants and Variations

A first observation is that in the above attack, all that matters is that we can consistently force the compression function to be the same permutation, block after block. Indeed, there was no need to pick 0^k as the block to repeat (and use as key to the blockcipher). Any other $2^n!$-fold repetition of an arbitrary k-bit string would have worked equally well.

A second observation is that the attack is not particular to $H(M,V) = E(M,V)$. Many other variants, such as $H(M,V) = M \oplus E(M,V)$ or $H(M,V) = M \oplus \pi(V + M)$ (related to the sponge construction [5, 6] and the JH construction [39]) work equally well. (Somewhat bizarly, in one looks at for instance Cubehash [4], which in each round XOR's in m bits of message followed by r repetitions of the same permutation, the message length of our collision reduces slightly for increasing r and decreasing m, contrary to the normal security tradeoff.)

Similar collisions can be found for any iterated compression function, despite the behaviour of iterating a random function F being slightly different than that of iterating a (random) permutation π. Indeed, consider the graph consisting of all chaining variables ($X \in \{0,1\}^n$) as nodes and an arc between X and Y iff $Y = \pi(X)$ (resp. $Y = F(X)$). In the case of a permutation, this graph consists of only cycles, in particular all nodes are on a cycle so taking the permutation to the least common multiple of all occurring cycle lengths (in the cycle decomposition of the permutation) results in the identity. For a random function however, a lot of nodes are actually not on cycles themselves, but rather lead up to them (cf. Pollard-ρ [29, 15]). To let the attack go through, we first need to walk to a cycle and, once we are on a cycle (with high probability), the attack works as before. Walking to a cycle can be done by prepending a sufficiently long string to both colliding messages, e.g., $\mathbf{M} = 1^{k(B+1)}0^L$ and $\mathbf{M}' = 1^{kB}0^L 1^k$, for integer B (even for $B = 0$ this gives a constant success probability over the choice of either the random function or the initial vector of the iteration).

A.2 Interpretation

It is doubtful whether the above "attack" has any relevance in practice: the message lengths involved are completely beyond anything remotely reasonable for practical applications (hence the name *unreal collision*). It does indicate that some care should be taken that the underlying permutation π does not have an exploitable cycle decomposition, but finding an otherwise reasonable looking permutation with bad cycle decomposition might actually be a fairly hard problem in itself. However, we did find the collisions useful in theory though, mainly to iron out certain subtleties in the formalization of security statements (models, definitions and theorems).