Attacking the Knudsen-Preneel Compression Functions*

Onur Özen[1,**], Thomas Shrimpton[2,***], and Martijn Stam[1]

[1] EPFL IC IIF LACAL, Station 14, CH-1015 Lausanne, Switzerland
{onur.ozen,martijn.stam}@epfl.ch
[2] Dept. of Computer Science, Portland State University, Room 120, Forth Avenue Building, 1900 SW 4th Avenue, Portland OR 97201, USA
teshrim@cs.pdx.edu

Abstract. Knudsen and Preneel (Asiacrypt'96 and Crypto'97) introduced a hash function design in which a linear error-correcting code is used to build a wide-pipe compression function from underlying blockciphers operating in Davies-Meyer mode. In this paper, we (re)analyse the preimage resistance of the Knudsen-Preneel compression functions in the setting of public random functions.

We give a new non-adaptive preimage attack, beating the one given by Knudsen and Preneel, that is *optimal* in terms of query complexity. Moreover, our new attack falsifies their (conjectured) preimage resistance security bound and shows that intuitive bounds based on the number of 'active' components can be treacherous.

Complementing our attack is a formal analysis of the query complexity (both lower and upper bounds) of preimage-finding attacks. This analysis shows that for many concrete codes the time complexity of our attack is optimal.

1 Introduction

Cryptographic hash functions remain one of the most used cryptographic primitives, and the design of provably secure hash functions (relative to various security notions) is an active area of research. From an appropriate perspective, most hash function designs can be viewed as the Merkle-Damgård iteration of a blockcipher-based compression function (where a single permutation can be regarded as a degenerate or fixed key blockcipher).

The classical PGV blockcipher-based compression functions [16] have an output size matching the blocksize n of the underlying blockcipher. Yet even for the optimally secure ones [1, 23], the (time) complexity of collision- and preimage-finding attacks is at most $2^{n/2}$, resp. 2^n; when $n = 128$ (e.g. AES) the resulting bounds have been deemed unacceptable for current practice.

* This work has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.
** Supported by a grant of the Swiss National Science Foundation, 200021-122162.
*** Supported by NSF grants CNS-0627752 and CNS-0845610.

S. Hong and T. Iwata (Eds.): FSE 2010, LNCS 6147, pp. 94–115, 2010.

This mismatch between desired output sizes for blockciphers versus hash functions has been recognized early on (dating back to Yuval [27]), and blockcipher-based compression functions that output more than n bits have been put forth. This output expansion is typically achieved by calling the blockcipher multiple times and then combining the resulting blockcipher outputs in some clever way. In the 1990s many so-called double-length constructions (where $2n$ bits are output) were put forth, but large classes of these were subsequently broken. Only recently have a few double-length constructions been supported by formal security proofs; see e.g. [5, 6, 7, 13, 15] for an overview. In any case, the standard approach in designing wider-output compression functions has been to fix a target output size (and often a target number of blockcipher calls as well) and then to build a compression function that is optimally collision-resistant for that size.

In three papers [8, 9, 10], Knudsen and Preneel adopted a different approach, namely to let the output size and (relatedly) the number of blockcipher calls vary as needed in order to guarantee a particular security target. Specifically, given r independent ideal compression functions $f_1, \ldots, f_r$, each mapping cn-bits to n bits, they create a new 'bigger' compression function outputting rn bits.[1]

The $f_1, \ldots, f_r$ are run in parallel, and each of their inputs is some linear combination of the blocks of message and chaining variable that are to be processed; the rn-bit output of their construction is the concatenation of the outputs of these parallel calls. The elegance of the KP construction is in *how* the inputs to $f_1, \ldots, f_r$ are computed. They use the generator matrix of an $[r, k, d]$ error-correcting code over $\mathbb{F}_{2^c}$ to determine how the ck input blocks of the 'big' compression function are xor'ed together to form the inputs to the underlying r functions. (In a generalization they consider the f_i as mapping from bcn' to bn' bits instead and use a code over $\mathbb{F}_{2^{bc}}$.)

Under a broad—but *prima facie* not unreasonable—assumption related to the complexity of finding collisions in parallel compression functions, Knudsen and Preneel show that any attack needs time at least $2^{(d-1)n/2}$ to find a collision in their construction. Thus for a code with minimum distance $d = 3$, one obtains a 2^n collision-resistance bound. For preimage resistance, Knudsen and Preneel *conjecture* that attacks will require at least $2^{(d-1)n}$ time. They also give preimage- and collision-finding attacks that, curiously, are mostly independent of the minimum distance. For preimage resistance the attacks of Knudsen and Preneel meet their conjectured bound (at least for MDS codes). For collision resistance the story is different, as for many of the codes they consider there is a considerable gap between the actual complexity of their attacks and their $2^{(d-1)n/2}$ bound. Watanabe [25] has subsequently shown a collision attack that is more efficient than Knudsen and Preneel's attack for many of their parameter sets. He was even able to show that the collision resistance *lower* bound given by Knudsen and Preneel is wrong for certain parameters (e.g. for $3 < d \leq k$).

[1] Note that Knudsen and Preneel also propose to instantiate the underlying ideal compression functions with a blockcipher run in Davies-Meyer mode and to iterate the compression function to obtain a full blockcipher-based hash function. See the full version for details.

Table 1. Knudsen-Preneel constructions (cf. [10, Table V]) based on $2n$-to-n bit primitive (PuRF or single-key blockcipher). Non-MDS parameters in italic.

Code	$sn + mn \to sn$	Preimage Resistance			
		Query Complexity	Our Attack Time	KP-Conj. Time	KP-Attack Time
$[r, k, d]_{2^e}$	$2kn \to rn$	$2^{rn/k}$	Sec. 5	$2^{(d-1)n}$	Thm. 1
$[5, 3, 3]_4$	$(5+1)n \to 5n$	$2^{5n/3}$	$2^{5n/3}$	2^{2n}	2^{2n}
$[\mathit{8}, \mathit{5}, \mathit{3}]_{\mathit{4}}$	$(8+2)n \to 8n$	$2^{8n/5}$	$2^{8n/5}$	2^{2n}	2^{3n}
$[\mathit{12}, \mathit{9}, \mathit{3}]_{\mathit{4}}$	$(12+6)n \to 12n$	$2^{4n/3}$	$2^{4n/3}$	2^{2n}	2^{3n}
$[\mathit{9}, \mathit{5}, \mathit{4}]_{\mathit{4}}$	$(9+1)n \to 9n$	$2^{9n/5}$	$2^{11n/5}$	2^{3n}	2^{4n}
$[\mathit{16}, \mathit{12}, \mathit{4}]_{\mathit{4}}$	$(16+8)n \to 16n$	$2^{4n/3}$	$2^{7n/3}$	2^{3n}	2^{4n}
$[6, 4, 3]_{16}$	$(6+2)n \to 6n$	$2^{3n/2}$	$2^{3n/2}$	2^{2n}	2^{2n}
$[8, 6, 3]_{16}$	$(8+4)n \to 8n$	$2^{4n/3}$	$2^{4n/3}$	2^{2n}	2^{2n}
$[12, 10, 3]_{16}$	$(12+8)n \to 12n$	$2^{6n/5}$	$2^{6n/5}$	2^{2n}	2^{2n}
$[9, 6, 4]_{16}$	$(9+3)n \to 9n$	$2^{3n/2}$	2^{2n}	2^{3n}	2^{3n}
$[16, 13, 4]_{16}$	$(16+10)n \to 16n$	$2^{16n/13}$	2^{2n}	2^{3n}	2^{3n}

Our contribution. This paper offers a new security analysis of the KP construction when the underlying compression functions are modeled as public random functions (PuRFs). In the process we also introduce a precise formalization of the Knudsen-Preneel transform and, more generally, blockwise-linear schemes. We directly address the conjectured preimage-resistance security by describing an attack taking into account both query and time-complexity; we see the latter as especially important when considering attacks. Our attacks go well *below* the conjectured lower bound by Knudsen and Preneel, demonstrating its incorrectness and, more generally, that intuition about security derived from the number of active functions can be misleading.

Our main result is a new preimage attack whose time and query complexity (ignoring the constant and logarithmic factors) is summarized[2] in Tables 1 and 2. From a practical point of view, the time complexity of our attack beats the one given by KP in every case but two, namely when the code is $[4, 2, 3]_8$ or $[5, 2, 4]_8$; in both cases we match the original attack, moreover we show that it is optimal for the former. Startlingly, in the $[12, 9, 3]_4$ case our preimage attack is *even faster than the collision attack* proposed by Knudsen and Preneel. So in that case we have uncovered a new collision attack as well!

Reducing the query complexity. We begin with the simple observation that $(0^a \,||\, x_1) \oplus (0^a \,||\, x_2)$ yields a string of the form $(0^a \,||\, X)$. More generally, any linear combination of strings with the same pattern of fixed zero bits will yield a string with the same form. By restricting the queries (to the PuRFs) to strings with the same (blockwise) pattern, we can optimize the *yield* (the maximum

[2] We note that our attack is not specific to the cases $c \in \{2, 3\}$, it also works for instance against the compression functions suggested by Knudsen and Preneel with $c = 5$ (mimicking the MD4 and MD5 situation).

Table 2. Knudsen-Preneel constructions (cf. [10, Table VIII]) based $3n$-to-n bit primitive (PuRF or double-key blockcipher).

Code	$sn + mn \to sn$	Preimage Resistance			
		Query Complexity	Our Attack Time	KP-Conj. Time	KP-Attack Time
$[r, k, d]_{2^e}$	$3kn \to rn$	$2^{rn/k}$	Sec. 5	$2^{(d-1)n}$	Thm. 1
$[4, 2, 3]_8$	$(4+2)n \to 4n$	2^{2n}	2^{2n}	2^{2n}	2^{2n}
$[6, 4, 3]_8$	$(6+6)n \to 6n$	$2^{3n/2}$	$2^{3n/2}$	2^{2n}	2^{2n}
$[9, 7, 3]_8$	$(9+12)n \to 9n$	$2^{9n/7}$	$2^{9n/7}$	2^{2n}	2^{2n}
$[5, 2, 4]_8$	$(5+1)n \to 5n$	$2^{5n/2}$	2^{3n}	2^{3n}	2^{3n}
$[7, 4, 4]_8$	$(7+5)n \to 7n$	$2^{7n/4}$	$2^{9n/4}$	2^{3n}	2^{3n}
$[10, 7, 4]_8$	$(10+11)n \to 10n$	$2^{10n/7}$	2^{2n}	2^{3n}	2^{3n}

number of compression function evaluations an adversary can make given a particular number of queries to the oraclized, ideal objects that underlie it). This observation allows us to reduce the query complexity of a preimage-finding attack to the bare minimum and also allows the attack to be deterministic and non-adaptive.

The results we derive here (Section 4) are relevant beyond the KP construction. In particular, they apply to all constructions in which the inputs to the blockciphers (or PuRFs) are determined by *blockwise* linear combinations of blocks of compression function input. This includes the schemes discussed by Peyrin et al. [15] and Seurin and Peyrin [20].

Exploiting the dual code to reduce the time complexity. When mounting our reduced-query attack against a KP construction with parameters $[r, k, d]_{2^e}$, the result is r lists with partial preimages (under each of the f_i respectively) and, with high probability, a full preimage is 'hiding' among these lists. That is to say, when we consider all possible combinations of partial preimages, some will correspond to a codeword and others will not. To reduce the time complexity we need to be able to find such a 'codeword' (being an actual, full preimage) efficiently among all possibilities.

The main innovation of our attack (Section 5) is in how to find full preimages from the lists of partial preimages. It is based on the observation that codewords in the dual code can be used to express relations between PuRF-inputs that correspond to a codeword. Using known techniques to solve the generalized birthday problem (see e.g. [2, 3, 19, 24]) this allows us to prune the lists and consequently find a preimage for the compression function faster (than a naive approach or than Knudsen and Preneel). In the full version we explore additional reductions in the memory requirements.

Proving optimality. A secondary result of this paper is a security proof for preimage resistance of the Knudsen-Preneel compression functions in the information-theoretic model. Here (Theorem 3) we determine a lower bound on the query complexity for a computationally unbounded adversary to successfully find

preimages. We give a concrete bound and, to interpret it, switch to an asymptotic assessment. This shows that the query complexity of our new attack is essentially optimal (up to a small factor).

Since the lower bounds on the query complexity serve as 'best case' lower bounds for the complexity of real-world attacks, we can conclude that our new preimage-finding attack is *optimal* whenever the time complexity of our attack matches its query complexity. This happens for 9 out of the 16 schemes: for the seven MDS schemes with $d = 3$, and for codes $[8, 5, 3]_4$ and $[12, 9, 3]_4$. For the remaining seven schemes we leave a gap between the information-theoretic lower bound and the real-life upper bound.

2 Preliminaries

Blockwise-linear compression functions. A *compression function* is a mapping $H : \{0,1\}^{tn} \to \{0,1\}^{sn}$ for some blocksize[3] $n > 0$ and integer parameters $t > s > 0$. For positive integers c and n, we let $\mathrm{Func}(cn, n)$ denote the set of all functions mapping $\{0,1\}^{cn}$ into $\{0,1\}^n$. A compression function is *PuRF-based* if its mapping is computed by a program with oracle access to a finite number of specified oracles $f_1, \ldots, f_r$, where $f_1, \ldots, f_r \xleftarrow{\$} \mathrm{Func}(cn, n)$. When a PuRF-based compression function operates on input W, we write $H^{f_1,\ldots,f_r}(W)$ for the resulting value. Of primary interest for us will be *single-layer* PuRF-based compression functions without feedforward. These call all oracles in parallel and compute the output based only on the results of these calls; in particular, input to the compression function is not further considered.

Most PuRF-based (and blockcipher-based) compression functions are of a special type. Instead of arbitrary pre- and postprocessing, one finds only functions that are blockwise linear. For example, consider all of the PGV hash functions. An advantage of a blockwise approach is that it yields simple-looking hash functions whose security is easily seen to be determined by the blocksize n. Linearity allows for relatively efficient implementation via bitwise exclusive-or of n-bit blocks. The Knudsen-Preneel construction is also blockwise linear, so let us define formally what is a blockwise-linear single-layer PuRF-based compression function without feedforward, an unwieldy name we shorten to *blockwise-linear scheme*.

Definition 1 (Blockwise-linear scheme). *Let r, c, b, t, s be positive integers and let matrices $\mathsf{C}^{\mathrm{PRE}} \in \mathbb{F}_2^{rcb \times tb}$, $\mathsf{C}^{\mathrm{POST}} \in \mathbb{F}_2^{sb \times rb}$ be given. We define $H = \mathsf{BL}^b(\mathsf{C}^{\mathrm{PRE}}, \mathsf{C}^{\mathrm{POST}})$ to be a family of single-layer PuRF-based compression functions $H_n : \{0,1\}^{tn} \to \{0,1\}^{sn}$, for all positive integers n with $b|n$. Specifically, let $n'b = n$, and $f_1, \ldots, f_r \in \mathrm{Func}(cn, n)$. Then on input $W \in \{0,1\}^{tn}$ (interpreted as column vector), ${H_n}^{f_1 \cdots f_r}(W)$ computes the digest $Z \in \{0,1\}^{sn}$ as follows:*

[3] We include the blocksize in the definition for convenience later on—it is not a necessity and only mildly restrictive.

1. *Compute* $X \leftarrow (\mathsf{C}^{\mathrm{PRE}} \otimes I_{n'}) \cdot W$;
2. *Parse* $X = (x_i)_{i=1\ldots r}$ *and for* $i = 1\ldots r$ *compute* $y_i = f_i(x_i)$;
3. *Parse* $(y_i)_{i=1\ldots r} = Y$ *and output* $Z = (\mathsf{C}^{\mathrm{POST}} \otimes I_{n'}) \cdot Y$.

where $\otimes$ *denotes the Kronecker product and* $I_{n'}$ *the identity matrix in* $\mathbb{F}_2^{n' \times n'}$.

In the definition above we silently identified $\{0,1\}^n$ with the vector space $\mathbb{F}_2^n$, etc. The map corresponding to $(\mathsf{C}^{\mathrm{PRE}} \otimes I_{n'})$ will occasionally be denoted C^{PRE}. It will be convenient for us to write the codomain of C^{PRE} as a direct sum, so we identify $\{0,1\}^{rcn}$ with $\bigoplus_{i=1}^{r} V_i$ where $V_i = \mathbb{F}_2^{cn}$ for $i = 1, \ldots, r$. If $x_1 \in V_1$ and $x_2 \in V_2$, then consequently $x_1 + x_2$ will be in $V_1 \oplus V_2$. (This extends naturally to $L_1 + L_2$ when $L_1 \subset V_1, L_2 \subset V_2$.) If we want to add 'normally' in $\mathbb{F}_2^{cn}$ we write $x_1 \oplus x_2$ which conveniently corresponds to exclusive or and the result will be in $\mathbb{F}_2^{cn}$ as expected.

Preimage resistance. A *preimage-finding adversary* is an algorithm with access to one or more oracles, and whose goal is to find a preimage of some specified compression function output. We will consider adversaries in two scenarios: the information-theoretic one and a more realistic concrete setting. For information-theoretic adversaries the only resource of interest is the number of queries made to their oracles. Otherwise, these adversaries are considered (computationally) unbounded. In the concrete setting, on the other hand, we are interested in the actual runtime of the algorithm (when fixing any reasonable computational model) and, to a lesser extent, its memory consumption (and code-size[4]). Without loss of generality, in both settings adversaries are assumed not to repeat queries to oracles nor to query an oracle outside of its specified domain.

There exist several definitions of preimage resistance, depending on the distribution of the element for which a preimage needs to be found. The strongest notion is that preimage resistance should hold with respect to any distribution, which can be formalized as everywhere preimage resistance [17].

Definition 2 (Everywhere preimage resistance). *Let* $c, r, s, t > 0$ *be integer parameters, and fix a blocksize* $n > 0$. *Let* $H\colon \{0,1\}^{tn} \to \{0,1\}^{sn}$ *be a PuRF-based compression function taking* r *oracles* $f_1, \ldots, f_r \in \mathrm{Func}(cn, n)$. *The everywhere preimage-finding advantage of adversary* $\mathcal{A}$ *is defined to be*

$$\mathsf{Adv}^{\mathsf{epre}}_H(\mathcal{A}) = \max_{Z \in \{0,1\}^{sn}} \Big\{ \Pr\Big[f_1 \ldots f_r \xleftarrow{\$} \mathrm{Func}(cn, n), (W') \leftarrow \mathcal{A}^{f_1 \cdots f_r}(Z) : Z = H^{f_1 \cdots f_r}(W') \Big] \Big\}$$

Define $\mathsf{Adv}^{\mathsf{epre}}_H(q)$ *and* $\mathsf{Adv}^{\mathsf{epre}}_H(t)$ *as the maximum advantage over all adversaries making at most* q *queries to each of their oracles respectively running in time at most* t.

[4] We force algorithms to read their own code so the runtime is naturally lower bounded by the code-size.

Linear error correcting codes. An $[r, k, d]_{2^e}$ linear error correcting code $\mathcal{C}$ is the set of elements (codewords) in a k-dimensional subspace of $\mathbb{F}_{2^e}^r$, where the minimum distance d is defined as the minimum Hamming weight (taken over all nonzero codewords in $\mathcal{C}$). The dual code $[r, r-k, d^\perp]_{2^e}$ is the set of all elements in the $r-k$-dimensional subspace orthogonal to $\mathcal{C}$ (with respect to the usual inner product), and its minimum distance is denoted $d^\perp$.

Not all parameter sets are possible, in particular $r \geq k$ (trivial) and the Singleton bound puts a (crude) limit on the minimum distance $d \leq r - k + 1$. Codes matching the Singleton bound are called maximum distance separable (MDS). The dual code of an MDS code is MDS itself as well, so $d^\perp = k + 1$.

An $[r, k, d]_{2^e}$ code $\mathcal{C}$ can be generated by a matrix $G \in \mathbb{F}_{2^e}^{k \times r}$, meaning that $\mathcal{C} = \{x \cdot G | x \in \mathbb{F}_{2^e}^k\}$ (using row vectors throughout). Without loss of generality, we restrict ourselves to systematic generator matrices, that is $G = [I_k | P]$ for $P \in \mathbb{F}_{2^e}^{k \times (r-k)}$ and I_k the identity matrix in $\mathbb{F}_{2^e}^{k \times k}$.

3 The Knudsen-Preneel Hash Functions

Knudsen and Preneel [8, 9] introduced a family of hash functions employing error correcting codes. (We use the journal version [10] as our frame of reference). Although their work was ostensibly targeted at blockcipher-based designs, the main technical thread of their work develops a transform that extends the range of an 'ideal' compression function (blockcipher-based, or not) in a manner that delivers some target level of security. As is nowadays typical, we understand an ideal compression function to be a PuRF. In fact, the KP transform is a special instance of a blockwise-linear scheme (Definition 1), in which the inputs to the PuRFs are determined by a linear code over a binary field with extension degree $e > 1$, i.e. $\mathbb{F}_{2^e}$, and with C^{POST} being the identity matrix over $\mathbb{F}_2^{rb \times rb}$ (corresponding to concatenating the PuRF outputs). The extension field itself is represented as a subring of the matrix ring (of dimension equalling the extension degree) over the base field. We formalize this by an injective ring homomorphism $\varphi : \mathbb{F}_{2^e} \to \mathbb{F}_2^{e \times e}$ and let $\bar{\varphi} : \mathbb{F}_{2^e}^{r \times k} \to \mathbb{F}_2^{re \times ke}$ be the component-wise application of φ and subsequent identification of $(\mathbb{F}_2^{e \times e})^{r \times k}$ with $\mathbb{F}_2^{re \times ke}$ (we will use $\bar{\varphi}$ for matrices over $\mathbb{F}_{2^e}$ of arbitrary dimensions).

Definition 3 (Knudsen-Preneel transform). *Let $[r, k, d]$ be a linear code over $\mathbb{F}_{2^e}$ with generator matrix $G \in \mathbb{F}_{2^e}^{k \times r}$. Let $\varphi : \mathbb{F}_{2^e} \to \mathbb{F}_2^{e \times e}$ be an injective ring homomorphism and let b be a positive divisor of e such that $ek > rb$. Then the Knudsen-Preneel compression function $H = \mathsf{KP}^b([r, k, d]_{2^e})$ equals $H = \mathsf{BL}^b(\mathsf{C}^{\text{PRE}}, \mathsf{C}^{\text{POST}})$ with $\mathsf{C}^{\text{PRE}} = \bar{\varphi}(G^T)$ and $\mathsf{C}^{\text{POST}} = I_{rb}$.*

If $H = \mathsf{KP}^b([r, k, d]_{2^e})$, then $H_n : \{0,1\}^{kcn} \to \{0,1\}^{rn}$ with $c = e/b$ is defined for all n for which b divides n. Moreover, H_n is based on r PuRFs in $\text{Func}(cn, n)$. For use of H in an iterated hash function, note that per invocation (of H) one can compress $(ck - r)$ message blocks (hence $ek > rb$ ensures actually compression is taking place), and the rate of the compression function is $ck/r - 1$.

We will concentrate on the case $(b, e) \in \{(1, 2), (2, 4), (1, 3)\}$ and then in particular on the 16 parameter sets given by Knudsen and Preneel. (Since b is uniquely determined given e, we will often omit it.) For an illustrative example of this formalism, please see Appendix A.

Knudsen and Preneel's security claims. Knudsen and Preneel concentrate on the collision resistance of their compression function in the complexity theoretic model. Under a fairly generous (but plausible) assumption, they essentially[5] show that if $H = \mathsf{KP}^b([r, k, d]_{2^e})$, then finding collisions in H_n takes time at least $2^{(d-1)n/2}$. The intuition behind this result is fairly simple. The use of a code of minimum distance d implies that for any pair of differing compression function inputs $W \neq W'$ there are at least d different PuRF inputs. That is, if $(x_i)_{i=1..r}$ and $(x'_i)_{i=1..r}$ are the respective PuRF inputs, then there is an index set $\mathcal{I} \subseteq \{1, \ldots, r\}$ such that $|\mathcal{I}| \geq d$ and for all $i \in \mathcal{I}$ it holds that $x_i \neq x'_i$. Thus, for W and W' to collide, one needs to find collisions for the PuRFs f_i for all $i \in \mathcal{I}$ simultaneously. Also, as the dimension of the code is k, there exist k PuRFs that can be attacked independently, say $f_1, ..., f_k$. Now, this is where their assumption comes into play. Namely, finding a collision is assumed to take $2^{vn/2}$ time where v is the number of PuRFs f_j, $j \in \{k+1, \ldots, r\}$, whose inputs satisfy $x_j \neq x'_j$ once $W \neq W'$. From the Singleton bound, one has $r - k \geq d - 1$. Hence, $v \geq d - 1$.

For preimage resistance Knudsen and Preneel do not give a corresponding theorem and assumption, yet they do conjecture it to be essentially the square of the collision resistance, that is, they conjecture that finding a preimage will take at least time $2^{(d-1)n}$.

Known attacks. To lowerbound the security of their construction, Knudsen and Preneel also present two attacks, one for finding preimages [10, Proposition 3] and one for finding collisions [10, Proposition 4]. We summarize these here, using our formalism.

Theorem 1 (Knudsen-Preneel attacks). *Let $H = \mathsf{KP}^b([r, k, d]_{2^e})$ be given and consider H_n (with b dividing n). Then*

1. *Preimages can be found in time $\max(2^{n(r-k)}, k2^{rn/k})$, using as many PuRF evaluations and requiring $ek2^{(r-k)n/k}$ n-bit blocks of memory;*
2. *Collisions can be found in time $\max(2^{n(r-k)/2}, k2^{(r+k)n/2k})$, using as many PuRF evaluations and requiring $ek2^{(r-k)n/2k}$ n-bit blocks of memory.*

4 Information-Theoretic Considerations

Bounding the yield. In the information-theoretic setting, the yield (Definition 4) of an adversary captures the number of compression function evaluations the adversary can make given the queries made so far. The concept has proven very fruitful in attacking schemes [12] and proving security [20, 21].

[5] Their actual theorem statements [10, Theorems 3 and 4] are phrased existentially.

Definition 4. *Let $H^{f_1,\ldots,f_r}$ be a compression function based on (ideal) primitives $f_1,\ldots,f_r$. The yield of an adversary after a set of queries to $f_1,\ldots,f_r$, is the number of inputs to H for which he can compute $H^{f_1,\ldots,f_r}$ given the answers to his queries. With* $\mathrm{yield}(q)$ *we denote the maximum expected yield given q queries to each of the oracles $f_1,\ldots,f_r$.*

For an arbitrary tn-to-sn bit compression function with r underlying cn-to-n primitives (each called once), the known lower bound on the yield [22, Theorem 6] is $\mathrm{yield}(q) \geq 2^{tn}(q/2^{cn})^r$. However, for the blockwise-linear schemes (Definition 1) it is possible to obtain a much bigger yield (being single-layer does not help either) and in particular it is *independent* of the number of primitive calls r.

Theorem 2. *Let $H = \mathsf{BL}^b(\mathsf{C}^{\mathrm{PRE}}, \mathsf{C}^{\mathrm{POST}})$ be a blockwise linear scheme with parameters c, t, s, r. Consider H_n with b dividing n. Then*

$$\mathrm{yield}(q) \geq 2^{\lfloor \frac{\lg q}{bc} \rfloor bt} \approx q^{t/c} \ .$$

Proof. Recall that t is the number of external n-bit input blocks and c the number of internal n-bit input blocks and that all n-bit blocks are subdivided into b n'-bit blocks. Set $n_q = \lfloor \lg q/(bc) \rfloor$ and $\mathcal{X} = (0^{n'-n_q} \times \{0,1\}^{n_q})^{bc}$. For each of the bc internal input subblocks, set the first $n' - n_q$ bits identical zero and let the rest range over all possibilities in $\{0,1\}^{n_q}$. All combinations of the internal input subblocks are combined (under concatenation) to give $(2^{n_q})^{bc} \leq q$ distinct inputs for any particular internal function. Query the f_i on these inputs (precisely corresponding to $\mathcal{X}$ defined above), for $i = 1,\ldots,r$.

Consider an external input W that consists of a concatenation of subblocks each with the first $n' - n_q$ bits set to zero. Then, due to linearity, $(\mathsf{C}^{\mathrm{PRE}} \otimes I_{n'}) \cdot W$ will map to a collection of PuRF-inputs all corresponding to the queries formed above, and hence this W will contribute to the yield. Since there are $2^{n_q bt}$ possible W that adhere to the format, we get the stated lower bound on the yield.

The approximation follows by ignoring the floor and simplifying the resulting expression. Although this can lead to slight inaccuracies, for increasing n there will be more and more values of q for which the expression is precise (so when $q = 2^{\alpha n}$ for rational α the expression is precise infinitely often). □

Information-theoretic attacks. Intuitively, when the yield for a tn-to-sn compression function gets close to $2^{sn/2}$, a collision is expected (birthday bound) and once it surpasses 2^{sn} a collision is guaranteed (pigeon hole) and a preimage expected. The bounds for permutation-based compression functions by Rogaway and Steinberger [18] are based on formalizing this intuition. In the claims below we relate what the bound on the yield implies for preimage and collision resistance, assuming that the yield results in more or less uniform values. Note that the assumption certainly does not hold in general (hence 'presumably'), see also the discussion below.

Claim (Consequences for blockwise-linear schemes). Let $H = \mathsf{BL}^b(\mathsf{C}^{\mathrm{PRE}}, \mathsf{C}^{\mathrm{POST}})$ be a blockwise linear scheme with parameters c, t, s, r. Consider H_n with b dividing n.

1. If $q \geq 2^{scn/t}$ then $\text{yield}(q) \geq 2^{sn}$ and a collision in H_n can be found with certainty; preimages can presumably be found with high probability.
2. If $q \geq 2^{scn/(2t)}$ then $\text{yield}(q) \geq 2^{sn/2}$ and collisions in H_n can presumably be found with high probability.

Information-theoretic security proof. The following result provides a security proof for preimage resistance of the Knudsen-Preneel compression functions in the information-theoretic model. That is, we give a lower bound on the query complexity (for a computationally unbounded adversary) of any preimage-finding attack. This bound shows that the query complexity of our new attack is optimal, up to a small factor. Therefore, the time complexity of our preimage attack is *optimal* whenever the time complexity of our attack matches its query complexity, and this is the case for 9 out of the 16 Knudsen-Preneel schemes.

Theorem 3. *Let $H = \mathsf{KP}^b([r,k,d]_{2^e})$ and, for b dividing n, consider H_n based on underlying PuRFs $f_i \in \text{Func}(cn,n)$ for $i=1,\ldots,r$ with $c = e/b$. Then for $q \leq 2^{cn}$ queries to each of the oracles and $\delta \geq 0$ an arbitrary real number:*

$$\mathsf{Adv}^{\mathsf{epre}}_H(q) \leq \frac{q^{1+\delta}}{2^{(r-k)n}} + p$$

where $p = \mathsf{Pr}\left[B[kq;2^{-n}] > kq^{(1+\delta)/k}\right]$ and $B[kq;2^{-n}]$ denotes a random variable counting the number of successes in kq independent Bernoulli trials, each with success probability 2^{-n}.

Proof (Sketch). Let $Z = z_1 \,||\, \cdots \,||\, z_r$ be the range point to be inverted. Recall that $f_1,\ldots,f_k$ are the functions corresponding to the systematic part of the $[r,k,d]_{2^e}$ code. Without loss of generality we will restrict our attention to an adversary $\mathcal{A}$ asking exactly q queries to each of its oracles and consider the transcript of the oracle queries and responses. Necessarily, in this transcript there is at least one tuple $(x_1,\ldots,x_k)$ of queries to $f_1,\ldots,f_k$ such that for all $i=1,\ldots,k$ we have $f_i(x_i) = z_i$. Notice that because $f_1,\ldots,f_k$ correspond to the systematic portion of the code, any tuple $(x_1,\ldots,x_k)$ of queries to these k PuRFs *uniquely* defines a tuple of queries $(x_{k+1},\ldots,x_r)$ to the remaining $r-k$ PuRFs. Thus the number of tuples $(x_1,\ldots,x_k)$ in the transcript such $f_i(x_i) = z_i$ for all $i=1,\ldots,k$ determines the number of tuples $(x_{k+1},\ldots,x_r)$ that could possibly be a (simultaneous) preimage for $z_{k+1} \,||\, \cdots \,||\, z_r$. Intuitively, if this number is bounded to be sufficiently small, then the probability that $\mathcal{A}$ could have won the epre game will also be small. Assuming $kq^{(1+\delta)/k}$ as an upperbound on the number of partial preimages for the systematic portion assures that at most $q^{(1+\delta)}$ tuples (by the arithmetic-geometric mean inequality) can possibly "work" for the non-systematic portion. So under that assumption, the probability of finding a preimage is at most $q^{1+\delta}/2^{(r-k)n}$. The bound follows. □

The following corollary makes the theorem more concrete. By considering a parameter δ that provides a good balance between the first and second terms in the bound, it follows that $\Omega(2^{rn/k})$ queries are necessary to win the epre

experiment. Since we already knew that $\mathcal{O}(2^{rn/k})$ queries are sufficient (under a reasonable uniformity assumption), this gives a complete characterization (up to increasingly small factors) of the query-complexity of finding preimages in Knudsen-Preneel construction.

Corollary 1. *Let $H = \mathsf{KP}^b([r,k,d]_e)$. Then asymptotically for n (with b dividing n) and $q \leq g(n)\left(\frac{2^n}{e}\right)^{r/k}$ with $g(n) = o(1)$:*

$$\mathsf{Adv}^{\mathsf{epre}}_H(q) = o(1) \ .$$

Proof (Sketch). Set $\delta = \frac{r(k-1)-k^2}{r}$ and substitute $q = g(n)\left(\frac{2^n}{e}\right)^{r/k}$ in the statement of Theorem 3. Substitution in the first term yields $\left(\frac{1}{e}\right)^{(r-k)}(g(n))^{1+\delta}$ which clearly vanishes whenever g does. For the second term in the bound of Theorem 3 we need to bound the tail probability of a binomial distribution. A standard Chernoff bound and substitution of the above δ and q gives that p vanishes as well for $g(n) = o(1)$. □

5 Preimage Attack against the Knudsen-Preneel Constructions

Setting the stage. Section 4 contains a theoretical attack with a minimal number of queries. This already allows us to turn Knudsen-Preneel's preimage attack from an adaptive one into a non-adaptive one and reduce its query consumption. In this section, we address reducing the time complexity as well.

Let $H = \mathsf{KP}^b([r,k,d]_{2^e})$ and n be given, where $b|n$ (and $bn' = n$ and $c = e/b$ as before). Consider a non-adaptive preimage-finding adversary $\mathcal{A}$ against H_n, trying to find a preimage for $Z \in \{0,1\}^{rn}$. For each $i = 1,\ldots,r$, $\mathcal{A}$ will commit to query lists $Q_i \subseteq V_i$ which, after querying, will result in a list of *partial* preimages $L_i = \{x_i \in Q_i | f_i(x_i) = z_i\}$. Since f_i is presumed random, we can safely assume that L_i is a set of approximately $|Q_i|/2^n$ randomly drawn elements of Q_i.

Finding a preimage then becomes equivalent to finding an element X in the range of C^{PRE} for which $x_i \in L_i$ for all $i = 1,\ldots,r$, or—exploiting the direct sum viewpoint—$X \in \sum_{i=1}^{r} L_i$. Due to the linearity of C^{PRE} at hand, the range-check itself is efficient for any given X, so a naive approach would be to simply exhaustively search $\sum_{i=1}^{r} L_i$. This would take time $|L|^r$.

An improvement can already be obtained by observing that, when a systematic matrix G is used to generate the code, any element $X_{1..k} \in \bigoplus_{i=1}^{k} V_i$ can uniquely (and efficiently) be extended to some X in the range of C^{PRE}. This lies at the heart of Knudsen and Preneel's adaptive attack and it can be adapted to the non-adaptive setting: for all $X_{1..k} \in \sum_{i=1}^{k} L_i$ compute its unique completion X and check whether for the remaining $i = k+1,\ldots,r$ the resulting $x_i \in L_i$. This reduces the time-complexity to $|L|^k$.

Organization. Still, we can do better. For concreteness, Section 5.1 provides a concrete warm-up example of our attack to the compression function $H =$

$\mathsf{KP}([5,3,3]_4)$. Section 5.2 builds on this and contains the core idea of how to reduce the time complexity of this new non-adaptive attack, as well as its application against compression functions based on MDS codes. The slightly more complicated non-MDS case is discussed in Section 5.3.

5.1 Example: Preimages in $\mathsf{KP}([5,3,3])_4$ in $\mathcal{O}(2^{5n/3})$ Time

Before describing our preimage attack in its full generality, we present an example of it applied to the compression function $H = \mathsf{KP}([5,3,3]_4)$.

Claim. For the compression function $H = \mathsf{KP}([5,3,3]_4)$, preimages in H_n can be found in $\mathcal{O}(2^{5n/3})$ time with a memory requirement of $\mathcal{O}(2^{4n/3})$ n-bit blocks.

Proof. We refer the reader to Appendix A for the details of G, φ and C^{PRE}. Let target digest $Z = z_1||\ldots||z_5$ be given, then our aim is to find the PuRF inputs $x_i = (x_i^1||x_i^2) \in \{0,1\}^{2n}$ such that $f_i(x_i) = z_i$ holds for all $i = 1,\ldots,5$, and $X = (\mathsf{C}^{\text{PRE}} \otimes I_{n'}) \cdot W$ for some compression function input W (where X is comprised of the five x_i). In this case W is a preimage for Z.

The attack starts with what we call the Query Phase. Namely, for each $i = 1,\ldots,5$ and for all $x_i^1, x_i^2 \in 0^{n/6} \times \{0,1\}^{5n/6}$, we query $f_i(x_i)$ and keep a list L_i of pairs that hit the target digest z_i. As a result, a total of $2^{5n/3}$ queries are made (in $2^{5n/3}$ time) per PuRF, resulting in $|L_i| \approx 2^{5n/3}/2^n = 2^{2n/3}$ (as each query has probability 2^{-n} to hit its target).

Since any tuple $(x_1, x_2, x_3) \in L_1 \times \ldots \times L_3$ uniquely determines a preimage candidate W, finding a preimage is equivalent to finding an element $X \in L_1 \times \ldots \times L_5$ in the range of C^{PRE}. To do this efficiently, we will first identify the tuples (x_1, x_2, x_3, x_4) in the lists that can be complemented (not necessarily using $x_5 \in L_5$) to an element in the range. From the generator matrix G it can be seen that this complementation is possible iff $x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0$. (The task is actually to determine whether a random vector y is a valid codeword; this can easily be detected by checking $y \cdot H^T = 0$ where H is the parity check matrix of the underlying code C.) So let us define

$$L_{\{1,2,3,4\}} = \{(x_1, x_2, x_3, x_4) \in L_1 \times L_2 \times L_3 \times L_4 \mid x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0\} \ .$$

We can construct $L_{\{1,2,3,4\}}$ efficiently using a standard technique related to the generalized birthday problem. It starts with the Merge Phase, where we create the lists $\tilde{L}_{\{1,2\}}$ and $\tilde{L}_{\{3,4\}}$ defined by

$$\tilde{L}_{\{1,2\}} = \{((x_1, x_2), x_1 \oplus x_2) \mid (x_1, x_2) \in L_1 \times L_2\}\,,$$
$$\tilde{L}_{\{3,4\}} = \{((x_3, x_4), x_3 \oplus x_4) \mid (x_3, x_4) \in L_3 \times L_4\}$$

both sorted on their second component. In the Join Phase we look for the collisions in their second components. Since $|L_i| \approx 2^{2n/3}$, creating either $\tilde{L}$ takes about $\mathcal{O}(n2^{4n/3})$ time and $\mathcal{O}(2^{4n/3})$ memory. (In general, the smallest $\tilde{L}$ is sorted and stored and the other is used for collision check.) Since $\tilde{L}_{\{1,2\}}$ and $\tilde{L}_{\{3,4\}}$ both have roughly $2^{4n/3}$ elements and they need to collide on $2n$ bits, of which $n/3$ bits

are set to zero, the expected number of collisions is about $(2^{4n/3})^2/2^{(2-1/3)n} = 2^n = |L_{\{1,2,3,4\}}|$.

We now have the collision list $L_{\{1,2,3,4\}}$ and all that needs to be done is to check, for each of its elements, whether the corresponding $x_5 \in L_i$. If this is the case, then (x_1, x_2, x_3) is a valid preimage. This final phase we call the FINALIZATION phase. It is clear that it cannot take much longer than it took to create $L_{\{1,2,3,4\}}$. Moreover, the expected number of preimages output is 1. Note that $|L_{\{1,2,3,4\}}| \approx 2^n$ and $|L_5| \approx 2^{2n/3}$. Again, we need to check the correspondence on $2n$ bits, of which $n/3$ are set to zero. Hence, we do expect to find $2^{(1+2/3)n}/2^{(2-1/3)n} = 1$ preimage.

Picking up the stepwise time and memory complexities gives the desired result. □

5.2 Generic Attack against MDS Schemes

Our attack on the compression function $\mathsf{KP}([5,3,3]_4)$ can be generalized to other Knudsen-Preneel compression functions. Note that the attack above consists of four steps:

1. QUERY PHASE to generate the lists of partial preimages;
2. MERGE PHASE where two sets of lists are each merged exhaustively;
3. JOIN PHASE where collisions between the two merged lists are selected resulting in fewer partial preimages that however are preimage of a larger part of the target digest.
4. FINALIZATION where the remaining partial preimages are filtered for being a full preimage.

(As a slight, standard optimization trick to save some memory one could generate and store one merged list only and, when creating the second merged list amortize with the JOIN and FINALIZATION phases. Further memory optimizations are explored in the full version.)

The core observation. From a high level, our approach is simple: we first identify an index set $\mathcal{I} \subseteq \{1, \ldots, r\}$ defining a subspace $\bigoplus_{i\in\mathcal{I}} V_i$ for which the range of C^{PRE} (when restricted to this subspace), is not surjective. By (blockwise-linear) construction, C^{PRE} will then map to a subspace of $\bigoplus_{i\in\mathcal{I}} V_i$ of at most dimension $(|\mathcal{I}| - 1)cn$ (over $\mathbb{F}_2$). As a consequence, we will be able to prune significantly the total collection of candidate preimages in $\sum_{i\in\mathcal{I}} L_i$, keeping only those elements that are possibly in the range of C^{PRE} restricted to $\bigoplus_{i\in\mathcal{I}} V_i$. In the following, we will show how to *efficiently* find an index set $\mathcal{I}$ and how to *efficiently* prune.

It turns out that an important parameter determining the runtime of our preimage attack is $d^\perp$, the minimum distance of the dual code. Let χ be the function that maps $h \in \mathbb{F}_{2^e}^r$ to the set of indices of non-zero entries in h. Thus, $\chi(h) \subseteq \{1, \ldots, r\}$ and $|\chi(h)|$ equals the Hamming weight of the codeword. If $h \in \mathcal{C}^\perp$, then for $\mathcal{I} = \chi(h)$ we have precisely the property that allows us to prune $\sum_{i\in\mathcal{I}} L_i$ for partial preimages. The following proposition develops the key

result for understanding our attack and the role the dual code plays in it. The interpretation follows the proposition.

Proposition 1. *Let* $H = \mathsf{KP}^b([r,k,d]_{2^e})$ *and* $M \in \mathbb{F}_2^{e\times re/b}$ *be given. Suppose that* $M = \bar{\varphi}(h^T)$ *for some* $h \in \mathbb{F}_{2^e}^r$*, then for all positive integers* n' *it holds that* $(M \otimes I_{n'}) \cdot (\mathsf{C}^{\text{PRE}} \otimes I_{n'}) \cdot W = 0$ *for all* $W \in \{0,1\}^{ken'}$ *iff* $h \in \mathcal{C}^\perp$.

Proof. Let $h \in \mathbb{F}_{2^e}^r$ and $W \in \{0,1\}^{ken'}$ be given. Let $M = \bar{\varphi}(h^T)$ and recall that $\mathsf{C}^{\text{PRE}} = \bar{\varphi}(G^T)$ where G is a generator of $\mathcal{C}$. Then

$$\begin{aligned}(M \otimes I_{n'}) \cdot (\mathsf{C}^{\text{PRE}} \otimes I_{n'}) \cdot W &= (\bar{\varphi}(h^T) \otimes I_{n'}) \cdot (\bar{\varphi}(G^T) \otimes I_{n'}) \cdot W \\ &= ((\bar{\varphi}(h^T) \cdot \bar{\varphi}(G^T)) \otimes I_{n'}) \cdot W \\ &= (\bar{\varphi}((Gh)^T) \otimes I_{n'}) \cdot W\end{aligned}$$

The statement that $(\bar{\varphi}((Gh)^T) \otimes I_{n'}) \cdot W = 0$ for all $W \in \{0,1\}^{ken'}$ is equivalent to the statement that $\bar{\varphi}((Gh)^T) = 0$. Since φ is injective, this in turn is equivalent to $(Gh)^T = 0$. By definition, it holds that $Gh = 0$ iff $h \in \mathcal{C}^\perp$. □

In essence, this proposition tells us that if we are given a codeword $h \in \mathcal{C}^\perp$ and an element $X \in \mathbb{F}_2^{rcn}$ (to be input to the PuRFs), then X can only be in the range of C^{PRE} if $(\bar{\varphi}(h^T) \otimes I_{n'}) \cdot X = 0$. Since the only parts of X relevant for this check are those lining up with the nonzero entries of h, we get that $\mathcal{I} = \chi(h)$ is the droid we are looking for. Indeed, an element $X \in \sum_{i\in\chi(h)} L_i$ can be completed to an element in the range of C^{PRE} iff $(\bar{\varphi}(h) \otimes I_{n'}) \cdot (X + 0) = 0$ (where we write $X + 0$ for embedding into the larger $\oplus_{i=1}^r V_i$).

Efficient creation of

$$L_h = \left\{ X \in \sum_{i\in\chi(h)} L_i \mid (\bar{\varphi}(h) \otimes I_{n'}) \cdot (X + 0) = 0 \right\}$$

is done adapting standard techniques [3, 19, 24] by splitting the codeword in two and looking for all collisions. Suppose that $h = h_0 + h_1$ with $\chi(h_0) \cap \chi(h_1) = \emptyset$, and define, for $j = 0, 1$

$$\tilde{L}_{h_j} = \left\{ (X_j, (\bar{\varphi}(h_j) \otimes I_{n'}) \cdot (X_j + 0)) \mid X_j \in \sum_{i\in\chi(h_j)} L_i \right\} .$$

Then L_h consists of the elements $X_0 + X_1$ for which $(X_0, Y_0) \in \tilde{L}_{h_0}, (X_1, Y_1) \in \tilde{L}_{h_1}$, and $Y_0 = Y_1$. By sorting the two $\tilde{L}$'s the time complexity of creating L_h is then roughly the maximum cardinality of the three sets $\tilde{L}_{h_0}, \tilde{L}_{h_1}$, and L_h involved. It therefore clearly pays dividends to minimize the Hamming weights of h_0 and h_1, which is done by picking a codeword $h \in \mathcal{C}^\perp$ of minimum distance $d^\perp$ and splitting it (almost) evenly.

For an MDS code, we know that $d^\perp = k + 1$. As a result, if h attains this, the map C^{PRE} is injective when restricted to $\bigoplus_{i\in\chi(h)} V_i$ (or else the minimum distance

Algorithm 1 (Preimage attack against MDS-based schemes).

Input: $H = \mathsf{KP}^b([r,k,d]_{2^e})$, block size n with $b|n$ and target digest $Z \in \{0,1\}^{rn}$.

Output: A preimage $W \in \{0,1\}^{tn}$ such that $H_n(W) = Z$.

1. Query Phase. Define
$$\mathcal{X} = (\{0\}^{\frac{n}{b}-\frac{rn}{ek}} \times \{0,1\}^{\frac{rn}{ek}})^e$$
and, for $i = 1,\ldots,r$ let $Q_i = \mathcal{X} \subset V_i$. Query f_i on all $x_i \in Q_i$. Keep a list L_i of all partial preimages $x_i \in Q_i$ satisfying $f_i(x_i) = y_i$.
2. First Merge Phase. Find a nonzero codeword $h \in \mathcal{C}^\perp$ of minimum Hamming weight $d^\perp$. Let $h = h_0 + h_1$ with $\chi(h_0) \cap \chi(h_1) = \emptyset$ and of Hamming weights $\lfloor d^\perp/2 \rfloor$ and $\lceil d^\perp/2 \rceil$ respectively. Create, for $j = 0, 1$
$$\tilde{L}_{h_j} = \left\{ (X_j, (\bar{\varphi}(h_j) \otimes I_{n'}) \cdot (X_j + 0)) \mid X_j \in \sum_{i \in \chi(h_j)} L_i \right\}$$
both sorted on their second component.
3. First Join Phase. Create L_h consisting exactly of those elements $X_0 + X_1$ for which $(X_0, Y_0) \in \tilde{L}_{h_0}, (X_1, Y_1) \in \tilde{L}_{h_1}$, and $Y_0 = Y_1$.
4. Finalization. For all $X \in L_h$ create the unique W corresponding to it and check whether it results in $x_i \in L_i$ for all $i = 1,\ldots,r$. If so, output W.

would be violated). Hence, we know that all possible preimages given *all* the lists L_i are represented by the partial preimages contained in $\tilde{L}_h$. We can finalize by simply checking for all elements in $\tilde{L}_h$ whether its unique completion to $X \in \bigoplus_{i=1}^r V_i$ corresponds to $x_i \in L_i$ for all $i = 1,\ldots,r$ (where checking for $i \in \chi(h)$ can be omitted). The complete preimage-finding algorithm is given in Algorithm 1.

Reinterpreting the example. Let us revisit our preimage attack example on $H = \mathsf{KP}([5,3,3]_4)$ to see how it fits within the general framework. In the example we more or less magically came up with the relation $x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0$. We can now appreciate that this constraint is really imposed by the dual codeword $h = (1\ 1\ 1\ 1\ 0)$. Thus our example corresponds to Algorithm 1 with $\chi(h_0) = \{1,2\}$ and $\chi(h_1) = \{3,4\}$ (leading to a completely even division).

Note that one can also perform the attack based on other dual codewords of minimum distance, for instance $h = (1\ w\ w^2\ 0\ 1)$. These two minimum distance dual codewords can easily be found based on the given systematic generator matrix $G = [I_k|P]$ of the original code. Namely, the dual codeword in the $(j-k)^{th}$ row of the corresponding generator matrix of the dual code $G^\perp = [P^T|I_{r-k}]$ is used as h to check the membership for the list L_j for $j > k$. (In general finding a minimum distance codeword might be more involved, but the dimensions are sufficiently small to allow exhaustive search.)

Analysis of the preimage attack. We proceed with the analysis of the generic preimage attack by providing the justifications of our claims and the overall time and memory complexities. We initially maintain $d^\perp$ in the expressions for future use (when discussing non-MDS codes). The proof of Theorem 4 (together with that of Theorem 5) is given in Appendix B.

Theorem 4. *Let $H = \mathsf{KP}^b([r,k,d]_{2^e})$ be given and let $d^\perp$ be the minimum distance of the dual code of $\mathcal{C}$. Suppose $\mathcal{C}$ is MDS and consider the preimage attack described in Algorithm 1 run against H_n using $q = 2^{rn/k}$ queries ($|Q_i| = 2^{rn/k}$). Then the expected number of preimages output equals one and the expectations for the internal list sizes are:*

$$|L_i| = 2^{\frac{(r-k)n}{k}} \ , \ |L_h| = 2^{\frac{(d^\perp(r-k)-r)n}{k}} \ , \ |\tilde{L}_{h_0}| = 2^{\lfloor \frac{d^\perp}{2} \rfloor \frac{(r-k)n}{k}} \ , \ |\tilde{L}_{h_1}| = 2^{\lceil \frac{d^\perp}{2} \rceil \frac{(r-k)n}{k}} \ .$$

The average case time and memory complexity (expressed in the number cn-bit blocks) of the algorithm is $\mathcal{O}(2^{\alpha n})$ and $\mathcal{O}(2^{\beta n})$ respectively where (substituting $d^\perp = k+1$)

$$\alpha = \max\left(\frac{r}{k}, \lceil\frac{k+1}{2}\rceil\left(\frac{r-k}{k}\right), r-k-1\right) \ , \quad \beta = \lfloor\frac{k+1}{2}\rfloor\left(\frac{r-k}{k}\right)$$

which for $d = 3$ simplifies to $\alpha = \frac{r}{k} = 1 + \frac{2}{k}$ and $\beta \leq \frac{k+1}{k}$.

In our attack, we set the number of queries as suggested by a yield-based bound. Hence, as long as this first querying phase is dominating, we know that our attack is optimal, as in the case for example against $\mathsf{KP}([5,3,3]_4)$. When the querying phase is not dominating (indicated by a gap between the time complexities of our attack and the lower bounds given in Tables 1 and 2) further improvements might be possible.

5.3 Extending the Attack to Non-MDS Constructions

For non-MDS codes we can try to mount the preimage attack given by Algorithm 1, but in the FINALIZATION we encounter a problem. Since $d^\perp < k+1$ for non-MDS codes, the map C^{PRE} restricted to $\oplus_{i \in \chi(h)} V_i$ is no longer injective and we can no longer reconstruct a unique W corresponding to some $X \in L_h$. There are two possible fixes to this problem. One is to simply merge as yet unused lists L_i into L_h until reconstruction does become unique. We will refer to this as Algorithm $1'$.

However, a more efficient approach is to perform a second stage of merging and joining. In Algorithm 2 we simply paste in extra MERGE and JOIN phases in order to maintain the low complexity. We have only included one extra merge-join phase for non-MDS codes. For the parameters proposed by Knudsen and Preneel, this will always suffice. For other parameters possibly extra merge-join phases are required before full rank is achieved, we did not investigate this.

Algorithm 2 (Preimage attack against non-MDS-based schemes).

Input: $H = \mathsf{KP}^b([r,k,d]_{2^e})$, block size n with $b|n$ and target digest $Z \in \{0,1\}^{rn}$.

Output: A preimage $W \in \{0,1\}^{tn}$ such that $H_n(W) = Z$.

1. Query Phase. As in Algorithm 1.
2. First Merge Phase. As in Algorithm 1.
3. First Join Phase. As in Algorithm 1.
4. Second Merge Phase. Find a codeword $h' \in \mathcal{C}^{\perp} \backslash \mathbb{F}_{2^e} h$ of minimum Hamming weight (possibly exceeding $d^{\perp}$). Let $h' = h'_0 + h'_1$ with $\chi(h'_0) \cap \chi(h'_1) = \emptyset$, $\chi(h'_1) \cap \chi(h) = \emptyset$, and of Hamming weights yet to be determined. Create
$$\tilde{L}_{h'_0} = \left\{ \left(X_0, (\bar{\varphi}(h'_0) \otimes I_{n'}) \cdot (X_0 + 0)\right) \mid X_0 \in L_h + \sum_{i \in \chi(h'_0) \backslash \chi(h)} L_i \right\}$$
$$\tilde{L}_{h'_1} = \left\{ \left(X_1, (\bar{\varphi}(h'_1) \otimes I_{n'}) \cdot (X_1 + 0)\right) \mid X_1 \in \sum_{i \in \chi(h'_1)} L_i \right\} .$$
5. Second Join Phase. Create $L_{h'}$ consisting exactly of those elements $X_0 + X_1$ for which $(X_0, Y_0) \in \tilde{L}_{h'_0}, (X_1, Y_1) \in \tilde{L}_{h'_1}$, and $Y_0 = Y_1$.
6. Finalization. For all $X \in L_{h'}$ create the unique W corresponding to it and check whether it results in $x_i \in L_i$ for all $i = 1, \ldots, r$. If so, output W.

Analysis of the attack. Although the addition of one extra round of MERGEing and JOINing sounds relatively simple, the analysis of it is slightly tedious, mainly because the first joining creates some asymmetry between the lists (that was not present before). We note that in Theorem 5 below the value i for which T_1 attains its minimum really only has a choice of two, but its algebraic optimization would not ease readability and obscure the underlying meaning. Note that for the memory analysis, we use a modified version of the algorithm, that is memory-optimized without aversely affecting the running time.

Because it is less clear from the theorem what the actual cardinalities will end up being (and consequently which step will be dominating), Table 3 summarizes the relevant quantities for the four non-MDS compression functions $\mathsf{KP}([r,k,d]_4)$ suggested by Knudsen and Preneel [10]. Only for the $[9,5,4]_4$ code the second stage dominates the overall runtime.

Theorem 5. *Let $[r,k,d] \in \{[8,5,3],[12,9,3],[9,5,4],[16,12,4]\}$ be given with a generator matrix G for $[r,k,d]_4$ (as given by Magma's BKLC routine); let $d^{\perp}$ be the minimum distance of the dual code of $\mathcal{C}$. For $H = \mathsf{KP}([r,k,d]_4)$ consider the preimage attack described in Algorithm 2 run against H_n using $q = 2^{rn/k}$ queries ($|Q_i| = 2^{rn/k}$). Then, the expected number of preimages output equals*

Table 3. Preimage attacks on the KP compression functions based on $2n \to n$ PuRFs and *non*-MDS-codes

Code		Cardinalities related to our attack						Overall		Alg. 1′
$[r,k,d]_{2^e}$	$d^\perp$	$\lvert Q_i\rvert$	$\lvert\tilde{L}_{h_0}\rvert$	$\lvert\tilde{L}_{h_1}\rvert$	$\lvert L_h\rvert$	$\max\left(\lvert\tilde{L}_{h'_0}\rvert,\lvert\tilde{L}_{h'_1}\rvert\right)$	$\lvert L_{h'}\rvert$	Time	Memory	Time
$[8,5,3]_4$	4	$2^{8n/5}$	$2^{6n/5}$	$2^{6n/5}$	$2^{4n/5}$	$2^{7n/5}$	2^n	$2^{8n/5}$	$2^{6n/5}$	2^{2n}
$[12,9,3]_4$	7	$2^{4n/3}$	2^n	$2^{4n/3}$	2^n	$2^{4n/3}$	2^n	$2^{4n/3}$	2^n	2^{2n}
$[9,5,4]_4$	4	$2^{9n/5}$	$2^{8n/5}$	$2^{8n/5}$	$2^{7n/5}$	$2^{11n/5}$	2^{2n}	$2^{11n/5}$	$2^{8n/5}$	2^{3n}
$[16,12,4]_4$	11	$2^{4n/3}$	$2^{5n/3}$	2^{2n}	$2^{7n/3}$	$2^{7n/3}$	2^{2n}	$2^{7n/3}$	$2^{5n/3}$	2^{3n}

one and the expectations for the internal list sizes are for the first merge-join are as before (see Theorem 4) and for the second merge-join phase

$$\max(|\tilde{L}_{h'_0}|,|\tilde{L}_{h'_1}|) \le 2^{T_1 n}\ ,\ \min(|\tilde{L}_{h'_0}|,|\tilde{L}_{h'_1}|) \le 2^{T_2 n}\ ,\ |L_{h'}| \le 2^{(r-k-2)n}$$

where

$$T_1 = \min_{i\in\{0,\ldots,k-d^\perp+1\}} \left(\max\{\frac{(k-d^\perp+2-i)(r-k)}{k}, \frac{((i+d^\perp)(r-k)-r)}{k}\}\right)$$

and $T_2 = r-k+\frac{r}{k}-2-T_1$. *The expected time complexity of the algorithm is a small constant multiple of*

$$\max\left(q, |\tilde{L}_{h_1}|, |L_h|, |\tilde{L}_{h'_0}|, |\tilde{L}_{h'_1}|, |L_{h'}|\right)$$

requiring expected memory around $\max\left(|\tilde{L}_{h_0}|, \min(|\tilde{L}_{h'_0}|, |\tilde{L}_{h'_1}|)\right)$ *(expressed in the number cn-bit blocks).*

Choice of code. Our attacks against the four non-MDS codes were based on the generator matrix given by Magma's BKLC routine. It is conceivable that different, non-equivalent codes perform differently under our attack. Most importantly, they might not have the same $d^\perp$ which will certainly change some of the cardinalities involved in our attack. Although this does not automatically means the attack becomes faster or slower, it is certainly a possibility. We note that there is a trivial bound $d^\perp \le k$ (or else the code would be MDS), but in none of the four cases we achieved this bound. Stronger bounds on $d^\perp$ might be possible by extending the recently developed primal-dual distance bounds [11] to the $\mathbb{F}_4$ setting.

6 Conclusion

In this paper, we provide a new security analysis of the KP construction by directly addressing its conjectured preimage-resistance security. Firstly, we describe an attack taking into account both query and time-complexities. Our

attacks demonstrate that the conjectured lower bound by Knudsen and Preneel is incorrect and exemplify that security bounds derived from the number of active functions can be misleading. Secondly, we determine a lower bound on the query complexity for a computationally unbounded adversary to successfully find preimages. This shows that the query complexity of our new attack is essentially optimal (up to a small factor). Moreover, we can conclude that the time complexity of our new preimage-finding attack is optimal for 9 out of the 16 schemes. For the remaining seven schemes we leave a gap between the information-theoretic lower bound and the real-life upper bound.

Acknowledgement. We thank the anonymous referees for their comments, in particular pointing out the work of Watanabe [25].

References

1. Black, J., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function constructions from PGV. In: Yung [26], pp. 320–335
2. Camion, P., Patarin, J.: The Knapsack Hash Function proposed at Crypto 1989 can be broken. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 39–53. Springer, Heidelberg (1991)
3. Chose, P., Joux, A., Mitton, M.: Fast correlation attacks: An algorithmic point of view. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 209–221. Springer, Heidelberg (2002)
4. Dunkelman, O. (ed.): FSE 2009. LNCS, vol. 5665. Springer, Heidelberg (2009)
5. Fleischmann, E., Gorski, M., Lucks, S.: On the security of Tandem-DM. In: Dunkelman [4], pp. 84–103
6. Fleischmann, E., Gorski, M., Lucks, S.: Security of cyclic double block length hash functions. In: Parker [14], pp. 153–175
7. Knudsen, L., Muller, F.: Some attacks against a double length hash proposal. In: Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 462–473. Springer, Heidelberg (2005)
8. Knudsen, L.R., Preneel, B.: Hash functions based on block ciphers and quaternary codes. In: Kim, K.-c., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 77–90. Springer, Heidelberg (1996)
9. Knudsen, L.R., Preneel, B.: Fast and secure hashing based on codes. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 485–498. Springer, Heidelberg (1997)
10. Knudsen, L.R., Preneel, B.: Construction of secure and fast hash functions using nonbinary error-correcting codes. IEEE Transactions on Information Theory 48(9), 2524–2539 (2002)
11. Matsumoto, R., Kurosawa, K., Itoh, T., Konno, T., Uyematsu, T.: Primal-dual distance bounds of linear codes with application to cryptography. IEEE Transactions on Information Theory 52(9), 4251–4256 (2006)
12. Nandi, M., Lee, W., Sakurai, K., Lee, S.: Security analysis of a 2/3-rate double length compression function in black-box model. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 243–254. Springer, Heidelberg (2005)
13. Özen, O., Stam, M.: Another glance at double-length hashing. In: Parker [14], pp. 176–201

14. Parker, M.G. (ed.): Cryptography and Coding 2009. LNCS, vol. 5921. Springer, Heidelberg (2009)
15. Peyrin, T., Gilbert, H., Muller, F., Robshaw, M.: Combining compression functions and block cipher-based hash functions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 315–331. Springer, Heidelberg (2006)
16. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
17. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy, B.K., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
18. Rogaway, P., Steinberger, J.: Security/efficiency tradeoffs for permutation-based hashing. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 220–236. Springer, Heidelberg (2008)
19. Schroeppel, R., Shamir, A.: A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain NP-complete problems. SIAM Journal on Computing 10, 456–464 (1981)
20. Seurin, Y., Peyrin, T.: Security analysis of constructions combining FIL random oracles. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 119–136. Springer, Heidelberg (2007)
21. Shrimpton, T., Stam, M.: Building a collision-resistant compression function from non-compressing primitives. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 643–654. Springer, Heidelberg (2008)
22. Stam, M.: Beyond uniformity: Better security/efficiency tradeoffs for compression functions. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 397–412. Springer, Heidelberg (2008)
23. Stam, M.: Block cipher based hashing revisited. In: Dunkelman [4], pp. 67–83
24. Wagner, D.: A generalized birthday problem. In: Yung [26], pp. 288–303
25. Watanabe, D.: A note on the security proof of Knudsen-Preneel construction of a hash function (unpublished manuscript) (2006), `http://csrc.nist.gov/groups/ST/hash/documents/WATANABE_kp_attack.pdf`
26. Yung, M. (ed.): CRYPTO 2002. LNCS, vol. 2442. Springer, Heidelberg (2002)
27. Yuval, G.: How to swindle Rabin. Cryptologia 3, 187–189 (1979)

A Illustrating the KP-Transform: $\mathsf{KP}^1([5,3,3]_{2^2})$

Consider the compression function $H = \mathsf{KP}([5,3,3]_4)$. This builds a $6n \to 5n$ compression function using five underlying PuRFs each mapping $2n \to n$ (so the rate is $(2 \cdot 3 - 5)/5 = 1/5$). The preprocessing function C^{PRE} of H is defined by a generator matrix G of the code $[5,3,3]_4$. Using the G proposed in [10] and defining φ by (that is also the one given in [10])

$$\varphi(0) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \varphi(1) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \varphi(w) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \text{ and } \varphi(w^2) = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}.$$

we get

$$G = \begin{pmatrix} 1\,0\,0\,1 & 1 \\ 0\,1\,0\,1 & w \\ 0\,0\,1\,1 & w^2 \end{pmatrix} \quad \text{and} \quad \mathsf{C}^{\text{PRE}} = \begin{pmatrix} 1\,0\,0\,0\,0\,0\,1\,0\,1\,0 \\ 0\,1\,0\,0\,0\,0\,0\,1\,0\,1 \\ 0\,0\,1\,0\,0\,0\,1\,0\,1\,1 \\ 0\,0\,0\,1\,0\,0\,0\,1\,0\,1 \\ 0\,0\,0\,0\,1\,0\,1\,0\,0\,1 \\ 0\,0\,0\,0\,0\,1\,0\,1\,1\,1 \end{pmatrix}.$$

Therefore, given $W \in \{0,1\}^{6n}$, H_n computes the digest $Z \in \{0,1\}^{5n}$ as follows:

1. Compute $X \leftarrow (\mathsf{C}^{\text{PRE}} \otimes I_n) \cdot W$;
2. Parse $X = (x_i)_{i=1\ldots5}$ and for $i = 1...5$ compute $y_i = f_i(x_i)$;
3. Parse $(y_i)_{i=1\ldots5} = Y$ and output $Z = (I_5 \otimes I_n) \cdot Y$, equivalently $Z = y_1||...||y_5$.

B Runtime Analysis (Proof of Theorems 4 and 5)

We will proceed step by step to prove our claims. The first three steps are common for the MDS and non-MDS case (where for MDS codes $d^\perp = k+1$). The remaining steps are treated separately. In the complexity estimations below, we concentrate on expected values and largely ignore (the effects of) polynomial factors in n (e.g. due to memory access). Throughout memory is measured in multiples of cn-bit blocks.

1. Query Phase. The time complexity of this step is simply $2^{rn/k}$ PuRF evaluations as $q = 2^{rn/k}$. Per L_i we need $q/2^n = 2^{(r-k)n/k}$ memory.

2. First Merge Phase. The main computational part of this step is the generation of the lists $\tilde{L}_{h_j}$ for $j = 0, 1$. The time required for generating $\tilde{L}_{h_0}$ and $\tilde{L}_{h_1}$ essentially equals their respective sizes, namely $|L_i|^{|\chi(h_0)|}$ and $|L_i|^{|\chi(h_1)|}$. We left i unspecified since all the L_i should be about the same size, namely $|L_i| \approx 2^{(r-k)n/k}$. Since by construction, $|\chi(h_0)| = \lfloor d^\perp/2 \rfloor$ and $|\chi(h_1)| = \lceil d^\perp/2 \rceil$, the relevant cardinalities become $2^{(r-k)\lfloor d^\perp/2 \rfloor n/k}$ and $2^{(r-k)\lceil d^\perp/2 \rceil n/k}$. This is clearly dominated by the latter.

3. First Join Phase. This step constructs L_h by finding collisions between $\tilde{L}_{h_0}$ and $\tilde{L}_{h_1}$ in their second components. Since $|\tilde{L}_{h_0}| \cdot |\tilde{L}_{h_1}| \approx 2^{d^\perp(r-k)n/k}$ and we are interested in collisions on rn/k bits, we have $|L_h| \approx 2^{(d^\perp(r-k)-r)n/k}$ (which equals $2^{(r-k-1)n}$ for MDS codes, given that $d^\perp = k+1$). Each colliding element can be forwarded directly to the next step eliminating a need to store L_h. Moreover, the collision search can be performed in conjunction with step 2 storing $\tilde{L}_{h_0}$ and checking (and processing) collisions on the fly when generating $\tilde{L}_{h_1}$. This way the memory requirements are reduced to $|\tilde{L}_{h_0}| \approx 2^{(r-k)\lfloor d^\perp/2 \rfloor n/k}$.

4. Second Merge Phase and 5. Second Join Phase (for non-MDS codes). For this scenario we restrict to the four codes suggested by Knudsen and Preneel. For the (chosen) systematic generator matrices we can always find (by inspection) $h, h' \in \mathcal{C}^\perp$ with the property that h has minimal weight, $\{1, \ldots, k\} \subset$

$\chi(h) \cup \chi(h')$ (so we reach reach full rank and can FINALIZE afterwards) and the number of $i \in \chi(h')$ for which $i \notin \chi(h)$ equals $k - d^{\perp} + 2$.

As a result, the relation defined by h' (and thus the second phase) will involve $k - d^{\perp} + 2$ 'fresh' lists L_i (those for which $i \notin \chi(h)$ and $i \in \chi(h')$) with $|L_i| = 2^{(r-k)n/k}$, as well as L_h, for which $|L_h| = 2^{(d^{\perp}(r-k)-r)n/k}$. Hence, regardless of the way of MERGEing and JOINing there will be

$$2^{\frac{(d^{\perp}(r-k)-r)n}{k}} \cdot 2^{\frac{(k-d^{\perp}+2)(r-k)n}{k}} = 2^{(r-k+\frac{r}{k}-2)n}$$

elements in total to be checked for collisions. As in the previous step, collisions will be searched for on rn/k bits. This leads to a list $L_{h'}$ of roughly $|L_{h'}| = 2^{(r-k-2)n}$ elements at the end of SECOND JOIN PHASE.

To minimize the complexity of the merging phase, we need to find the sets $\chi(h'_0)$ and $\chi(h'_1)$ such that $\chi(h'_0) \cap \chi(h'_1) = \emptyset$ and the full $2^{(r-k+\frac{r}{k}-2)n}$ elements (involved in the merging) are distributed as evenly as possible *without* violating the constraints imposed by the asymmetric list sizes.

Suppose $|\tilde{L}_{h'_0}| = 2^{\alpha n}$ and $|\tilde{L}_{h'_1}| = 2^{\beta n}$ for α and β to be determined. The condition $\chi(h) \cap \chi(h'_1) = \emptyset$ implies that L_h is assigned to h'_0; assume that i further fresh L_i are used for h'_0. This automatically means that $|\chi(h'_1)| = (k - d^{\perp} + 2 - i)$ and furthermore that $|\tilde{L}_{h'_1}| = 2^{(k-d^{\perp}+2-i)(r-k)n/k}$ and $|\tilde{L}_{h'_0}| = 2^{i(r-k)n/k+(d^{\perp}(r-k)-r)n/k}$, implying $\alpha = ((d^{\perp}+i)(r-k)-r)/k$. Given a particular i, the merging time will be governed by the maximum of α and β whereas the storage requirement is similarly the minimum of that pair. In order to optimize the overall time complexity, we take the minimum (of the maximum just mentioned) over all i and denote the value by T_1 and, for the value i used, denote by T_2 the corresponding 'memory'-minimum. Note that $T_1 + T_2 = r - k + \frac{r}{k} - 2$. Collision finding can then be performed in $2^{T_1 n}$ time with a memory requirement of roughly $2^{T_2 n}$.

6. FINALIZATION. For each element in L_h (resp. in $L_{h'}$ for non-MDS codes) we need to perform a simple check (that we assume costs unit time and constant memory). For MDS codes, after the FIRST JOIN PHASE, we have that L_h has size roughly $|L_h| = 2^{(r-k-1)n}$. For the non-MDS case, we have already shown that $|L_{h'}| = 2^{(r-k-2)n}$ (at least for the four non-MDS codes provided by Knudsen and Preneel).

Picking up the obtained complexities for the various steps gives the desired overall complexity. □