Faster 2-regular information-set decoding

Daniel J. Bernstein[1], Tanja Lange[2], Christiane Peters[2], and Peter Schwabe[3]

[1] Department of Computer Science
University of Illinois at Chicago, Chicago, IL 60607–7045, USA
djb@cr.yp.to
[2] Department of Mathematics and Computer Science
Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, Netherlands
tanja@hyperelliptic.org, c.p.peters@tue.nl
[3] Institute of Information Science
Academia Sinica, 128 Section 2 Academia Road, Taipei 115-29, Taiwan
peter@cryptojedi.org

Abstract. Fix positive integers B and w. Let C be a linear code over $\mathbf{F}_2$ of length Bw. The 2-regular-decoding problem is to find a nonzero codeword consisting of w length-B blocks, each of which has Hamming weight 0 or 2. This problem appears in attacks on the FSB (fast syndrome-based) hash function and related proposals. This problem differs from the usual information-set-decoding problems in that (1) the target codeword is required to have a very regular structure and (2) the target weight can be rather high, so that there are many possible codewords of that weight.

Augot, Finiasz, and Sendrier, in the paper that introduced FSB, presented a variant of information-set decoding tuned for 2-regular decoding. This paper improves the Augot–Finiasz–Sendrier algorithm in a way that is analogous to Stern's improvement upon basic information-set decoding. The resulting algorithm achieves an exponential speedup over the previous algorithm.

Keywords: Information-set decoding, 2-regular decoding, FSB, binary codes.

1 Introduction

The FSB family of hash functions was submitted to the SHA-3 competition by Augot, Finiasz, Gaborit, Manuel, and Sendrier in 2008 [1]. The submission proposes six specific hash functions: FSB_{160}, FSB_{224}, FSB_{256}, FSB_{384}, FSB_{512}, and a toy version FSB_{48}. The index specifies the size of the output. The hash function consists of a compression function, which is iterated to compress the entire message one block at a time, and a hash function to handle the output of

This work was supported by the National Science Foundation under grant 0716498, by the European Commission under Contract ICT-2007-216499 CACE, and by the European Commission under Contract ICT-2007-216646 ECRYPT II. Permanent ID of this document: `c6c2347b09f3864994aefae5f5b6e7be`. Date: 2011.03.09.

the final round of the compression function. The designers chose Whirlpool as the final hash function.

The compression function is what gives this class of functions its name "fast syndrome-based hash functions". The compression function uses a matrix H over $\mathbf{F}_2$ of size $r \times 2^b w$, viewed as having w blocks of size $r \times 2^b$; the parameters here are, e.g., $r = 640$, $b = 14$, $w = 80$ for FSB_{160}. The matrix H is a parity-check matrix for a code of length $2^b w$ and dimension at least $2^b w - r$. A single iteration of the compression function takes as input a bit string of length bw, interprets the bit string as a sequence of w numbers $m_1, m_2, \ldots, m_w$ in $[0, 2^b - 1]$, and computes the sum of the columns indexed by $m_1 + 1, m_2 + 1, \ldots, m_w + 1$ in blocks $1, 2, \ldots, w$ respectively. The output of the compression function is therefore Hy, the syndrome of the vector $y = ((2^{m_1})_2, (2^{m_2})_2, \ldots, (2^{m_w})_2)$, where $(2^{m_i})_2$ means the 2^b-bit binary representation of 2^{m_i} in little-endian notation. The primary goal of the compression function is to make it difficult for attackers to find a collision, i.e., two distinct inputs that compress to the same output.

Details about how the matrix is constructed and how the message blocks are chained can be found in the design document [1] and in the papers [2], [3], [15], and [14] describing preliminary FSB designs. In [7] we proposed a more efficient family of syndrome-based hash functions called RFSB (for "really fast syndrome-based" hashing); RFSB differs from FSB in the parameter choices and in the way the matrix is constructed. For this paper the matrix-construction details do not matter; for stating the algorithms we consider a general $r \times 2^b w$ matrix, or even more generally an $r \times Bw$ matrix. We use FSB_{160} as an example to illustrate the ideas and the improvements in various algorithms.

Two distinct vectors y and y' having the same syndrome $Hy = Hy'$ do not necessarily correspond to a collision in the compression function, because not every vector can be written in the form $((2^{m_1})_2, (2^{m_2})_2, \ldots, (2^{m_w})_2)$. If the vectors y and y' correspond to colliding messages then they must have Hamming weight exactly 1 in each block. The sum $y + y'$ is then a nonzero 2-regular codeword, where *2-regular* means that the word has weight 0 or 2 in each block. Note that the concept of 2-regularity for $\mathbf{F}_2^{Bw}$ depends implicitly on the partitioning of Bw positions into w blocks; sometimes we write *w-block 2-regularity.*

Conversely, any 2-regular codeword can be written trivially as $y + y'$, where y and y' each have weight exactly 1 in each block. Any nonzero 2-regular codeword therefore immediately reveals a collision in this compression function. The problem of finding a collision is thus equivalent to the problem of *2-regular decoding*, i.e., the problem of finding a nonzero 2-regular codeword in a code, specifically the code defined by the parity-check matrix H.

There is an extensive literature on algorithms to search for low-weight words in codes. One can use any of these algorithms to search for a codeword of weight $2w$ (or of weight in $\{2, 4, 6, \ldots, 2w\}$), and then hope that the resulting codeword is 2-regular. However, it is better to pay attention to 2-regularity in the design of the low-weight-codeword algorithm. Augot, Finiasz, and Sendrier introduced the first dedicated 2-regular-decoding algorithm in the same 2003 paper [2] that introduced FSB and the 2-regular-decoding problem.

This paper generalizes and improves the Augot–Finiasz–Sendrier algorithm. The new algorithm combines ideas from various improved versions of low-weight information-set decoding, and restructures those ideas to fit the more complicated context of 2-regular codewords. In particular, our attack adapts ideas of Lee–Brickell, Leon, and Stern (see Section 2) to increase the chance of success per iteration at the expense of more effort per iteration. The increase in success chance outweighs the extra effort by an exponential factor.

Section 5 shows the impact of the new algorithm upon FSB_{48}, FSB_{160}, FSB_{256}, and RFSB-509. In each case our algorithm uses far fewer operations than the algorithm of [2]. Note, however, that all of these compression functions are conservatively designed; our algorithm is not fast enough to violate the security claims made by the designers.

All of these algorithms can be generalized to decoding arbitrary syndromes for codes over arbitrary finite fields $\mathbf{F}_q$. The only case that arises in our target application is decoding syndrome 0 over $\mathbf{F}_2$.

Model of computation. Like previous papers on information-set decoding, this paper counts the number of bit operations used for arithmetic, and ignores the cost of memory access. We have made no attempt to minimize the amount of memory used by our new algorithm, and we do not claim that our algorithm is an improvement over previous algorithms in models of computation that penalize memory access.

Other approaches. Information-set decoding is not the only strategy for finding 2-regular codewords. Three other attack strategies have been applied to the FSB collision-finding problem: linearization, generalized birthday attacks, and reducibility. See our survey [7, Section 4] for credits, citations, and corrections.

Information-set decoding, linearization, and generalized birthday attacks are generic techniques that apply to practically all matrices H. Reducibility is a special-purpose technique that relies on a particular structure of H (used in the FSB proposals from [15]) and that is easily combined with the generic techniques when it is applicable. The generic techniques have not been successfully combined with each other, and it is not clear that any one of these techniques is superseded by the others. Linearization seems unbeatable when w/r is not much below $1/2$, but it degrades rapidly in performance as w/r decreases. For small w/r the best technique could be information-set decoding or generalized birthday attacks. The FSB paper [3] says that generalized birthday attacks are a larger threat; the FSB submission [1, Table 4, "best attacks known": "collision search" column] says that information-set decoding is a larger threat; both of the underlying analyses are disputed in [4]. We recommend continuing investigation of all of these approaches.

2 Low-weight information-set decoding

This section reviews several improvements in low-weight information-set decoding, as background for our improvements in 2-regular information-set decoding.

Types of decoders. We systematically phrase the algorithms here as syndrome-decoding algorithms using parity-check matrices. The goal is to find a low-weight error vector e matching a given syndrome s for a given parity-check matrix H: specifically, to find $e \in \mathbf{F}_2^n$ with $\mathrm{wt}(e) \leq t$ such that $He = s$, given $s \in \mathbf{F}_2^r$ and $H \in \mathbf{F}_2^{r \times n}$. We abbreviate $n - r$ as k.

These algorithms can be, and in the literature often are, translated into word-decoding algorithms using generator matrices. The distinction between syndrome decoding and word decoding is minor: an application that wants word decoding can begin with a word $v \in \mathbf{F}_2^n$, compute the syndrome $s = Hv$, apply a syndrome-decoding algorithm to find e, and finally compute $v - e$, a codeword whose distance from v is at most t. The distinction between parity-check matrices and generator matrices is more important, and can have a noticeable effect on the efficiency of the algorithms, although we are not aware of any systematic study of this effect. It is typical in code-based cryptography for k to be larger than $n/2$, so parity-check matrices are smaller than generator matrices; this is particularly obvious for the parameters n and k appearing in FSB.

Plain information-set decoding. Information-set decoding was first suggested by Prange in [22] and was later used by McEliece [21] to estimate the security of code-based cryptography.

One iteration of plain information-set decoding works as follows. Select a random set of r columns of the $r \times n$ parity check matrix H of the code. Permute columns to move these to the right-hand side of the matrix, producing a matrix H'. Compute the inverse U of the rightmost $r \times r$ submatrix of H'; if the submatrix is not invertible — i.e., if the k non-selected columns are not an information set — then the iteration fails. If $\mathrm{wt}(Us) \leq t$ then the iteration has successfully found a low-weight error vector $e' = (0 \ldots 0|Us)$ matching Us for $UH' = (\cdots|I_r)$ and therefore matching s for H'; reversing the column permutation on the positions of e' produces a low-weight error vector e matching s for H. Otherwise the iteration fails. The method succeeds if there are no errors located in the information set.

As mentioned in the introduction, our main concern in this paper is the case $s = 0$. Prange's algorithm is not interesting for $s = 0$: its only possible output is 0. Weight-t codewords can be viewed as weight-t error vectors (relative to codeword 0) but will not be found by this algorithm (except in the degenerate case $t = 0$). The improved algorithms discussed below do not have this limitation, and remain interesting in the case $s = 0$: they allow sums of columns of I_r to be cancelled by sums of other columns of UH', so they can find nonzero error vectors having syndrome 0. A different way to handle syndrome 0 is to extend Prange's algorithm to scan the entire kernel of the $r \times r$ submatrix of H', allowing the submatrix to be non-invertible; this is the starting point for the algorithm of [2] discussed in the next section.

The standard improvements. Lee and Brickell in [18] improved Prange's method by choosing a small parameter $p \leq t$ and allowing p errors in the information set (together with $\leq t - p$ errors in the selected columns). This means

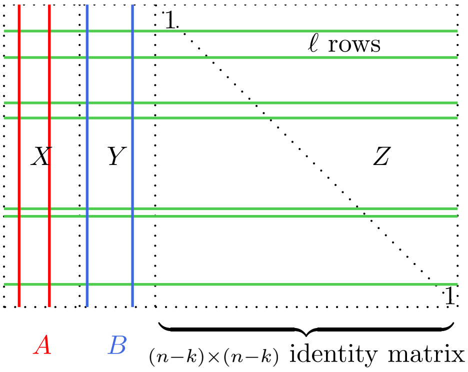

Fig. 2.1. One check in Stern's algorithm

checking $\binom{k}{p}$ combinations of columns but amortizes the costs of Gaussian elimination across those combinations.

Leon in [19], independently of Lee and Brickell, suggested p errors in the information set together with ℓ-row early aborts. Instead of checking the weight of the sum of each set of p columns, this algorithm checks only the sets that add up to 0 on a subset Z of the rows, where Z has size ℓ. The effort for each of the $\binom{k}{p}$ combinations is reduced from an r-row sum to an ℓ-row sum (plus an $(r-\ell)$-row sum with probability about $1/2^{\ell}$), at the cost of missing error vectors that have errors in the ℓ columns corresponding to Z.

The next year, in [24], Stern suggested the same improvements together with a collision speedup. The information set is partitioned into two sets X and Y. The Lee–Brickell parameter p is required to be even and is split as $(p/2)+(p/2)$. The algorithm searches for error vectors that have $p/2$ errors among the positions in X and $p/2$ errors among the positions in Y; all $(p/2)$-column subsets A of X are tested for matches with all $(p/2)$-column subsets B of Y. The algorithm follows Leon's by checking only the pairs (A,B) that add up to 0 on a subset Z of the rows. The collision speedup is as follows: instead of trying all pairs (A,B), the algorithm computes one list of Z-sums of $(p/2)$-column subsets A, and a second list of Z-sums of $(p/2)$-column subsets B, and then efficiently finds collisions between the two lists.

Figure 2.1 displays one checking step in Stern's algorithm for $p=4$ and $\ell=7$. The two leftmost solid columns (red on color displays) form the set A; the two rightmost solid columns (blue) form the set B. The pair (A,B) is considered only if these columns match on the ℓ positions indicated by solid rows (green), i.e., sum up to 0 on each of those positions. If so, the sum is computed on the full length r. If the sum has weight $t-4$ then the algorithm has found a word of weight t. This word has nonzero entries in the 4 columns indexed by A,B and the positions where the $t-4$ errors occur.

Further improvements. Many papers have proposed improvements to Stern's algorithm, for example in how the matrices H' and UH' are computed; in how the choices of columns in X and Y are handled; and in how the full test is done once a choice was successful on the ℓ positions. The most recent papers are [5], [16], and [6]; see those papers for surveys of previous work.

3 The Augot–Finiasz–Sendrier algorithm for 2-regular decoding

This section discusses the Augot–Finiasz–Sendrier algorithm [2, Section 4.2] for 2-regular decoding. The algorithm inputs are positive integers r, B, w and a parity-check matrix $H \in \mathbf{F}_2^{r \times n}$, where $n = Bw$. If the algorithm terminates then it outputs a nonzero w-block 2-regular codeword; recall that this means a nonzero codeword v such that, for each $i \in \{1, 2, \ldots, w\}$, the ith B-bit block of v has Hamming weight 0 or 2.

The algorithm can be generalized to decoding arbitrary syndromes, but we focus on syndrome 0 as discussed in the introduction. We refer to the positions of nonzero entries in the target codeword as error positions, viewing the target codeword as an error vector relative to codeword 0.

Review of the algorithm. Each iteration of this algorithm selects a set of r out of the n positions of columns from H, performs $r \times r$ Gaussian elimination to compute the kernel of those r columns, and checks each nonzero element of the kernel for 2-regularity. The selection is split evenly among w_0 blocks of H, where $w_0 \in \{1, 2, 3, \ldots, w\}$ is an algorithm parameter; assume for the moment that w_0 divides r and that $r/w_0 \le B$.

Out of all 2^r vectors supported in these r positions, only $\sum_{1 \le i \le w_0} \binom{r}{2i}$ have weight in $\{2, 4, 6, \ldots, 2w_0\}$, and only $\left(\binom{r/w_0}{2} + 1\right)^{w_0} - 1$ are nonzero 2-regular vectors. Each of these nonzero 2-regular vectors has, under suitable randomness assumptions on H, probability $1/2^r$ of being a codeword, so the expected number of codewords found by one iteration of the algorithm is $\left(\left(\binom{r/w_0}{2} + 1\right)^{w_0} - 1\right)/2^r$.

Augot, Finiasz, and Sendrier conclude that the success probability of an iteration is $\left(\binom{r/w_0}{2} + 1\right)^{w_0}/2^r$. Here they are ignoring the -1 above, and ignoring the difference between the success probability and the expected number of codewords (i.e., ignoring the possibility of an iteration finding two codewords at once), but these are minor effects.

The expected number of kernel elements is (again under suitable randomness assumptions on H, which we now stop mentioning explicitly) a constant; a large kernel occurs with only small probability. The bottleneck in the iteration is Gaussian elimination, using $O(r^3)$ bit operations.

Non-divisibility. Augot, Finiasz, and Sendrier say that for $w < \alpha r$ it is best to take $w_0 = w$; here $\alpha \approx 0.24231$ is chosen to maximize $\left(\binom{1/\alpha}{2} + 1\right)^{\alpha}$. Many of the published FSB parameters (r, B, w) have w dividing r and $w < \alpha r$; in all of these cases, w_0 will also divide r.

However, Augot, Finiasz, and Sendrier also consider many parameters with $w > \alpha r$, and say that in this case it is best to take $w_0 = \alpha r$. Presumably this means choosing w_0 as an integer very close to αr, but for almost all values of r this means that w_0 cannot divide r. This non-divisibility causes various problems that are not discussed in [2] and that invalidate some of the algorithm analysis in [2], as we now show.

The obvious way to interpret the algorithm to cover this case is to take some blocks with $\lfloor r/w_0 \rfloor$ selected columns, and some with $\lceil r/w_0 \rceil$, for a total of r columns. Write $f = \lfloor r/w_0 \rfloor$, $b = r - w_0 f = r \bmod w_0$, and $a = w_0 - b$; then one can take a blocks each with f columns and b blocks each with $f+1$ columns, for a total of r columns in w_0 blocks. For example, if $w_0 = 0.24r$, then one can take $5w_0 - r = 0.20r$ blocks with 4 columns and $r - 4w_0 = 0.04r$ blocks with 5 columns.

The number of 2-regular words supported in these r columns is exactly $\left(\binom{f}{2}+1\right)^a \left(\binom{f+1}{2}+1\right)^b$. Here we are counting, for each of the a blocks, the number of ways to choose 0 or 2 out of f columns; and, for each of the b blocks, the number of ways to choose 0 or 2 out of $f+1$ columns. The expected number of nonzero 2-regular codewords found by one iteration is thus

$$\left(\left(\binom{f}{2}+1\right)^a \left(\binom{f+1}{2}+1\right)^b - 1\right) \Big/ 2^r.$$

If $r/5 < w_0 \le r/4$ then this number is $(7^a 11^b - 1)/2^r = (7^{5w_0 - r} 11^{r-4w_0} - 1)/2^r \approx ((11/14)(7^5/11^4)^{w_0/r})^r$; e.g., approximately $2^{-0.300r}$ for $w_0 = \alpha r$.

The analysis in [2] incorrectly applies the formula $\left(\binom{r/w_0}{2}+1\right)^{w_0}/2^r$ without regard to the question of whether r/w_0 is an integer, and in particular for the case $w_0 = \alpha r$. This overstates the success probability by a small but exponential factor, for example claiming success probability $2^{-0.298r}$ for $w_0 = \alpha r$. The discrepancy is larger for ratios r/w_0 that are farther from integers; see Figure 3.1. Many of the curves plotted in [2] and [3, Section 4.4] need to be adjusted accordingly. This analysis also shows that the correct cutoff for w_0 is $0.25r$, not αr. We are not aware of any sensible algorithm modification that would rescue the analysis in [2].

Comparison to low-weight decoding. We emphasize that finding a nonzero 2-regular codeword is much harder than merely finding a word of weight $2w$. Each nonzero 2-regular codeword has weight in $\{2, 4, 6, \ldots, 2w\}$, but the opposite is very far from being true: most words of this weight will not have the right distribution of nonzero entries.

For example, consider FSB_{160}, with $r = 640$, $B = 2^{14}$, and $w = 80$. The parity check matrix is a $640 \times n$ matrix where $n = 2^{14} \cdot 80 = 1310720$. The Augot–Finiasz–Sendrier algorithm picks 640 columns in a regular pattern by taking 8 columns of each of the 80 blocks; by linear algebra identifies all codewords supported in those columns; and checks 2-regularity of each nonzero codeword. The number of nonzero 2-regular vectors supported in those columns is $(\binom{8}{2}+1)^{80} - 1 \approx 2^{388.64}$, so the number of nonzero 2-regular codewords supported in these columns is expected to be approximately $2^{388.64}/2^{640} = 2^{-251.36}$.

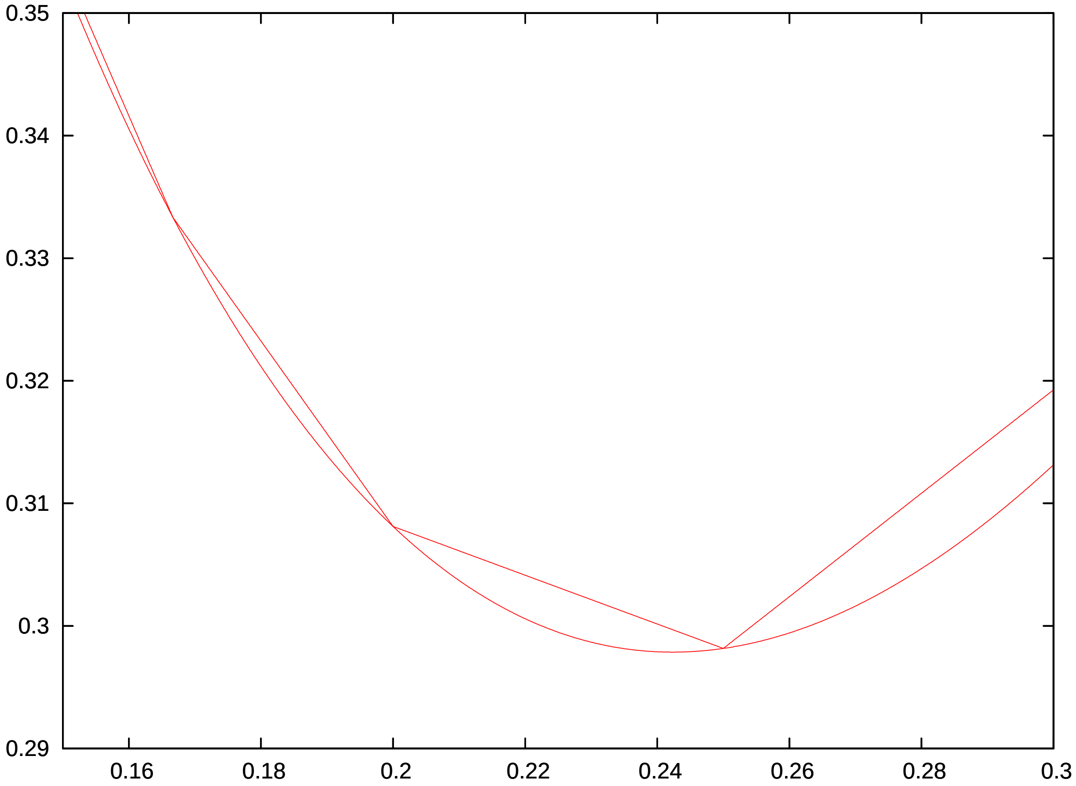

Fig. 3.1. Vertical axis, bottom curve: $y = 1 - x\log_2\left(\binom{1/x}{2}+1\right)$; the Augot–Finiasz–Sendrier algorithm was claimed to use asymptotically 2^{yr} iterations if $x = w_0/r$. Vertical axis, top segments: $y = 1-(x(f+1)-1)\log_2\left(\binom{f}{2}+1\right)-(1-xf)\log_2\left(\binom{f+1}{2}+1\right)$ where $f = \lfloor 1/x \rfloor$; the algorithm actually uses asymptotically 2^{yr} iterations if $x = w_0/r$.

For comparison, the number of weight-160 vectors supported in the same columns is approximately $\binom{640}{160} \approx 2^{514.44}$, so the number of weight-160 codewords supported in these columns is expected to be approximately $2^{514.44}/2^{640} = 2^{-125.56}$. The probability is slightly larger if weights $158, 156, \ldots$ are also allowed. This change in success criteria does not noticeably slow down the iteration, and it drastically reduces the number of iterations required for success.

This difference is even more extreme for $w \approx r/4$. Finding a w-block 2-regular codeword then takes an exponential number of iterations, approximately

$$2^r \bigg/ \left(\binom{4}{2}+1\right)^{r/4} = (16/7)^{r/4},$$

while finding a weight-$2w$ codeword takes a polynomial number of iterations, approximately

$$2^r \bigg/ \binom{r}{r/2} \approx 2^r \left(\sqrt{2\pi r/2}\left(\frac{r/2}{e}\right)^{r/2}\right)^2 \bigg/ (\sqrt{2\pi r}\left(\frac{r}{e}\right)^r) = \sqrt{\pi r/2}.$$

In both cases each iteration takes polynomial time.

4 A new algorithm for 2-regular decoding

This section presents a new algorithm for 2-regular decoding. This algorithm combines the standard improvements in low-weight decoding (see Section 2) with the Augot–Finiasz–Sendrier algorithm described in the previous section.

Impact of the improvements. One might guess that these improvements actually have very little effect, if any at all, upon the complexity of 2-regular decoding. There are several arguments for this guess:

- In the usual context of low-weight decoding, the standard improvements are often viewed as rather small. For example, Augot, Finiasz, and Sendrier in [2, Section 4.1] say that [10], [18], and [24] merely "reduce the degree of the polynomial part" of the complexity of information-set decoding.
- Plain information-set decoding and the Augot–Finiasz–Sendrier algorithm apply linear algebra to an $r \times r$ matrix. The improved algorithms by Lee–Brickell, Leon, and Stern start by inverting an $r \times r$ matrix but then have to multiply the inverse U by the remaining $r \times k$ submatrix of the parity-check matrix H. This extra multiplication has comparable cost to the inversion if k and r are on the same scale, as they usually are in applications of low-weight decoding; but k is usually much larger than r in applications of 2-regular decoding. For example, recall that FSB_{160} has $r = 640$ and $k = 1310080$.
- Speedups in low-weight decoding algorithms are usually aimed at the case of small weight (at most the Gilbert–Varshamov bound), where one expects to find at most one codeword. Normally 2-regular decoding is applied for much larger weights, where many solutions exist. There is no reason to think that speedups in one context should be effective for the other.

But there are also counterarguments. One can show that Stern's speedup is superpolynomial when parameters are properly optimized, and that the cost of linear algebra inside Stern's algorithm is asymptotically negligible. See [8]. For the same reasons, the cost of multiplying U by H is also negligible.

To firmly settle the arguments we show that our new algorithm for 2-regular decoding is faster than the old algorithm by an exponential factor. For example, for $w/r = 0.1435$, the number of bit operations in the new algorithm is $2^{0.2825r}$ times a polynomial factor (provided that $B \geq 2^8$), while the number of bit operations in the old algorithm is $2^{0.3621r}$ times a polynomial factor.

We also evaluate the polynomial factors in the operation count for the new algorithm. The next section considers various specific hash functions, showing in each case the concrete speedup from the Augot–Finiasz–Sendrier algorithm to our algorithm.

The new algorithm. Each iteration of this algorithm works as follows. Select a set of r out of the n positions of columns from H. Split the selection almost evenly (see below) among w_0 blocks of H, where $w_0 \in \{1, 2, 3, \ldots, w\}$ is an algorithm parameter with $r \leq Bw_0$.

The new algorithm will do more work than the old algorithm in this iteration: it will search for 2-regular codewords that have exactly 2ш errors in the $n-r$ non-selected positions. Here $\text{ш} \in \{1, \ldots, \lfloor w_0/2 \rfloor\}$ is another algorithm parameter. Note that the presence of 2ш errors will exclude the possibility of uselessly finding codeword 0.

Use Gaussian elimination to see whether the r selected vectors are linearly independent. This occurs with probability approximately 29%; see [11]. If the vectors are dependent, start the iteration over with a new selection of r positions; even with this restarting, Gaussian elimination is not a bottleneck for large ш. An alternative is to consider each kernel element, but for simplicity and speed we enforce linear independence.

The set of non-selected positions is now an information set for H, and the set of selected positions is the corresponding redundancy set. Gaussian elimination has also revealed a linear transformation that converts the selected r vectors into an $r \times r$ identity matrix; apply the same transformation to all of H. This uses quite a few operations, but it is not a bottleneck for large ш.

Assume for simplicity that w_0 is even. Partition the w_0 selected blocks of H into $w_0/2$ "left" blocks and $w_0/2$ "right" blocks; the codewords found by the algorithm will have exactly ш information-set errors spread among exactly ш left blocks, and exactly ш information-set errors spread among exactly ш right blocks. Also choose a set S of ℓ out of the r row indices, where ℓ is another algorithm parameter. This set S corresponds, via the $r \times r$ identity matrix, to ℓ elements of the redundancy set; the codewords found by the algorithm will have 0 errors in those positions, as in Stern's algorithm.

Build a list L as follows. Choose ш left blocks, choose one information-set position in each block, and add the S-indexed bits of those ш vectors, obtaining an ℓ-bit vector. Store the resulting ℓ-bit vector along with the chosen positions as the first entry in L. Repeat until L has N elements, where N is another algorithm parameter. Similarly build a list R of N elements, using the right blocks.

Find all ℓ-bit collisions between L and R. For each collision, compute the sum of the non-S-indexed bits of those 2ш vectors. If the sum has weight $2i - 2\text{ш}$ for some $i \in \{2\text{ш}, 2\text{ш} + 1, \ldots, w_0\}$ then it can be written trivially as a sum of $2i-2\text{ш}$ vectors from the redundancy set. The positions of those vectors, together with the positions from L and R, form a codeword of weight $2i$. Output the codeword if it is 2-regular.

Algorithm analysis. Write ℓ as $\ell_1 w_0 + \ell_0$ with $0 \le \ell_0 < w_0$. Our algorithm analysis assumes that $r-\ell$ is a multiple of w_0, say $f w_0$. In this case the algorithm can, and we assume that it does, select columns as follows: ℓ_0 blocks each contain exactly $f + \ell_1 + 1$ elements of the redundancy set, including $\ell_1 + 1$ elements corresponding to elements of S; $w_0 - \ell_0$ further blocks each contain exactly $f + \ell_1$ elements of the redundancy set, including ℓ_1 elements corresponding to elements of S. Note that each of these w_0 blocks contains exactly f non-S elements of the redundancy set.

This appears to be the most nicely balanced case of the algorithm. However, we do not assert that it is always optimal. The algorithm does not require w_0 to divide $r - \ell$; if w_0 does not divide $r - \ell$ then one can choose sizes so that the total matches and some sets have one extra element each, as in the previous section. We exclude this possibility solely to simplify the analysis.

An element of L and an element of R together specify a pattern of 2ш errors in 2ш blocks in the information set. For each of those blocks there are exactly f ways to choose a non-S element of the redundancy set within the block. For each of the $w_0 - 2\text{ш}$ remaining blocks there are exactly $1 + \binom{f}{2}$ ways to choose zero or two non-S elements of the redundancy set.

Putting together all of these choices produces a nonzero 2-regular error pattern. Each of the 2ш initially specified blocks contains exactly one error in the information set and exactly one non-S error in the redundancy set. Each of the $w_0 - 2\text{ш}$ remaining blocks contains exactly zero or exactly two non-S errors in the redundancy set. If this error pattern is a codeword then it will be found by the algorithm.

The expected number of codewords obtainable in this way is

$$\delta = \frac{N^2}{2^r} f^{2\text{Ш}} \left(1 + \binom{f}{2}\right)^{w_0 - 2\text{Ш}}$$

under suitable randomness assumptions. The factor N^2 counts the number of pairs of elements of L and elements of R, and the factor $1/2^r$ is the chance of an error pattern being a codeword. The success probability of an iteration is approximately $1 - \exp(-\delta)$.

The cost of an iteration is the cost of linear algebra, plus $2N\ell$ additions for the elements of L and R (assuming reuse of additions as in [5, Section 4]), plus approximately $(N^2/2^\ell)2\text{ш}(r - \ell)$ additions to handle the partial collisions. We could use early aborts as in [5] to reduce the factor $r - \ell$ here, but for simplicity we avoid doing so.

Parameter selection. For various choices of (r, B, w) we performed computer searches to identify good algorithm parameters $(w_0, \text{ш}, N, \ell)$. See Section 5 for examples. The search space is small enough that no general recommendations for parameter choices are needed, but we nevertheless make a few comments regarding the optimal parameters.

There are obvious benefits to increasing N in this algorithm: δ grows quadratically with N, while the cost of the iteration grows only linearly with N (assuming $N < 2^\ell$; see below). But there are two hard limits upon this benefit. First, N cannot exceed the number of choices of entries in L; this number is between $\binom{w_0/2}{\text{Ш}}(B - 1 - r/w_0)^{\text{Ш}}$ and $\binom{w_0/2}{\text{Ш}}(B + 1 - r/w_0)^{\text{Ш}}$. Second, the quadratic growth of δ with N stops producing a quadratic growth in the success probability of the iteration as δ approaches and passes 1, i.e., as N^2 approaches and passes $2^r f^{-2\text{Ш}} \left(1 + \binom{f}{2}\right)^{2\text{Ш} - w_0}$. Our computations suggest that, in general, the best operation counts for this algorithm balance the first and second limits: most of the possibilities for L and R are used, and a single iteration has a high chance of success.

If N is allowed to grow beyond 2^ℓ then the cost of the iteration begins to grow quadratically. One can compensate for this by increasing ℓ. In general it seems best to choose ℓ close to $\log_2 N$, as in previous information-set-decoding algorithms, so that the cost of building lists L and R is balanced with the costs of doing the full-length checks. Increasing ℓ has the disadvantage of directly reducing δ, but this appears to be outweighed by the advantage of keeping $N^2/2^\ell$ under control. Beware that increasing ℓ also has a disadvantage not visible in the bit-operation cost model: efficient collision detection needs about 2^ℓ bits of memory.

Asymptotics. Fix a positive integer B and a positive real number W. Assume that $w/r \to W$ as $r \to \infty$. The following analysis optimizes choices of positive real numbers $W_0, \text{Ш}, T$ for the following goals: if the algorithm parameters $w_0, \text{ш}, N, \ell$ are chosen so that $w_0/r \to W_0$, $\text{ш}/r \to \text{Ш}$, $(\log_2 N)/r \to T$, and $\ell/r \to T$, then the algorithm run time also satisfies $(\log_2 \text{time})/r \to T$; furthermore, T is as small as possible.

We impose several constraints upon the choices of $W_0, \text{Ш}, T$:

- The ratio $f = (1-T)/W_0$ is a positive integer. This allows ℓ and w_0 to be chosen so that $(r-\ell)/w_0 = f$ once r is sufficiently large. We suspect that our algorithm achieves smaller exponents without this constraint, but as noted above our algorithm analysis requires w_0 to divide $r-\ell$.
- $W_0 \le W$. This allows w_0 to be chosen in $\{2,4,6,\ldots,2\lfloor w/2\rfloor\}$.
- $\text{Ш} \le W_0/2$. This allows ш to be chosen in $\{1,2,3,\ldots,\lfloor w_0/2\rfloor\}$.
- $2T - 1 + 2\text{Ш}\log_2 f + (W_0 - 2\text{Ш})\log_2(1 + f(f-1)/2) = 0$; in other words, $(\log_2 \delta)/r \to 0$. This ensures that the algorithm succeeds within $2^{o(r)}$ iterations. We do not have a proof that this constraint is always optimal, but we impose it here to simplify the asymptotic analysis.
- $T \le (W_0/2)\log_2(W_0/2) - \text{Ш}\log_2 \text{Ш} - (W_0/2 - \text{Ш})\log_2(W_0/2 - \text{Ш}) + \text{Ш}\log_2(B-1-1/W_0)$. This ensures that N can be chosen below $\binom{w_0/2}{\text{Ш}}(B-1-r/w_0)^{\text{Ш}}$.

Under these constraints, the cost of an iteration is within a polynomial factor of $2^{(T+o(1))r}$, so the total number of bit operations used by the algorithm is also within a polynomial factor of $2^{(T+o(1))r}$.

We view this constrained optimization problem as a series of separate problems: one with $f = 1$, one with $f = 2$, one with $f = 3$, etc. For any particular $f < 1/W_0$, substituting $T = 1 - fW_0$ into $2T - 1 + 2\text{Ш}\log_2 f + (W_0 - 2\text{Ш})\log_2(1 + f(f-1)/2) = 0$ produces an equation for Ш in terms of W_0, namely

$$\text{Ш} = \frac{1 - (2f - \log_2(1 + f(f-1)/2))W_0}{2\log_2(1 + f(f-1)/2) - 2\log_2 f}.$$

If this does not satisfy $0 < \text{Ш} \le W_0/2$ then f and W_0 are incompatible. The final inequality $T \le \cdots$ then puts a lower bound on B. To summarize, each choice of W_0 has a finite list of possibilities for f, with each possibility immediately dictating Ш, T, and a lower bound on B.

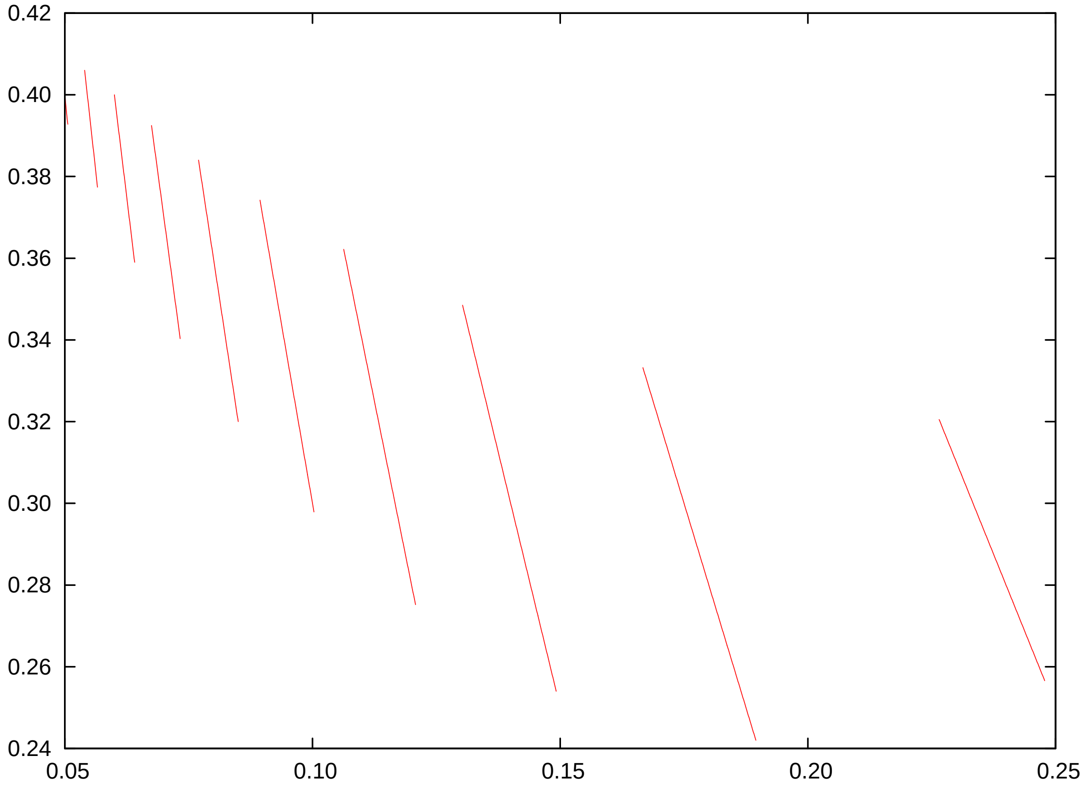

Fig. 4.1. T as a function of W_0. See text for description.

Figure 4.1 shows values of T that can be achieved in this way; Figure 4.2 shows the corresponding lower bounds on $\log_2 B$, and Figure 4.3 shows the corresponding Ш. The horizontal axis in each case is W_0. The values of f are, from right to left, first 3, then 4, etc. The figures omit values of W_0 that require $B > 2^{20}$.

For example, $W_0 = 0.2453$ and $f = 3$ produce $T = 0.2641$, $\text{Ш} \approx 0.022649$, and the lower bound $B \geq 2^8$. As another example, $W_0 = 0.1435$ and $f = 5$ produce $T = 0.2825$, $\text{Ш} \approx 0.027001$, and the lower bound $B \geq 2^8$. The smallest value of T in Figure 4.1 is $T = 0.2420$, achieved for $W_0 = 0.1895$, $f = 4$, $\text{Ш} \approx 0.009905$, and $B \geq 2^{19.81}$.

Given W we scan through $W_0 \leq W$ to minimize T. For example, any $W \geq 0.1435$ can use $T = 0.2825$ by taking $W_0 = 0.1435$, if $B \geq 2^8$. Figure 4.4 plots the resulting T as a function of W, and for comparison plots the exponent of the Augot–Finiasz–Sendrier algorithm.

Further improvements. Finiasz and Sendrier [16] improved Stern's algorithm by avoiding the static split into left and right in the information set. We adapt this approach to the situation of finding 2-regular words as follows. Build one list L by repeatedly picking Ш blocks and then for one column per block computing the sum on the ℓ positions specified by S. This increases the maximal value of N to approximately $\binom{w_0}{\text{Ш}}(B - r/w_0)^{\text{Ш}}$. Then search for ℓ-bit collisions within L. If a

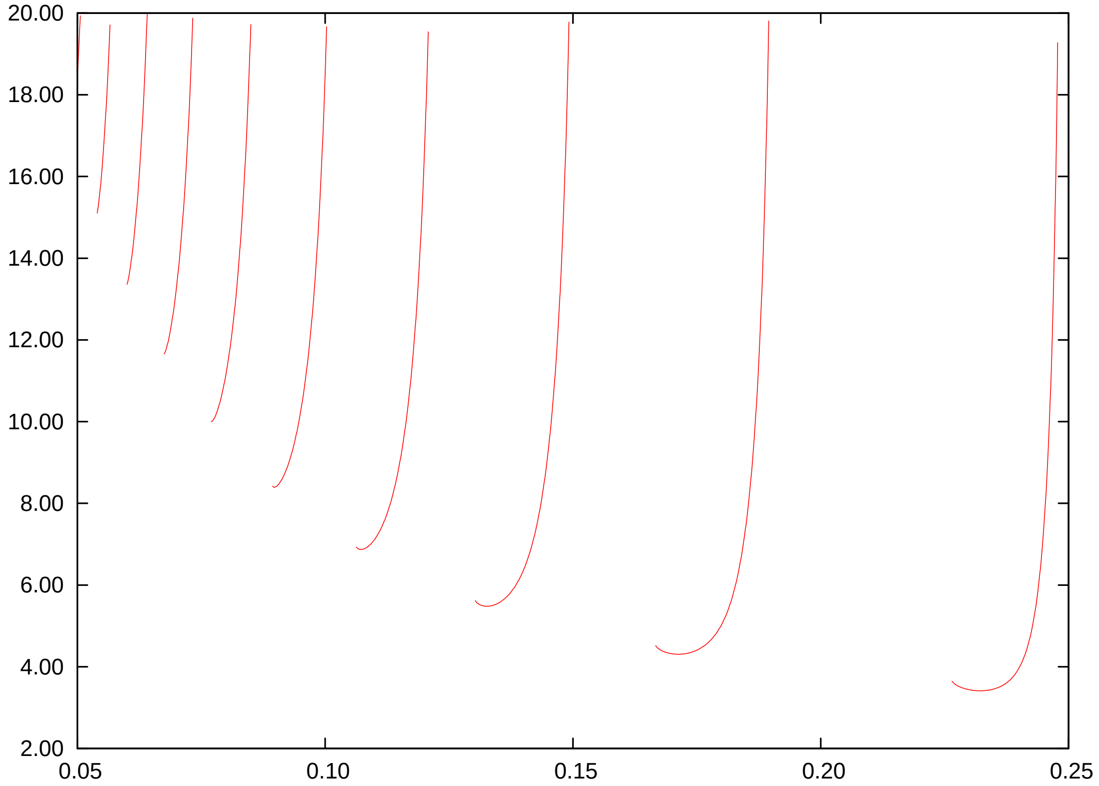

Fig. 4.2. Minimum $\log_2 B$ as a function of W_0. See text for description.

collision on the ℓ positions happens to involve j positions shared between the two entries from L the algorithm can still produce a 2-regular vector. The condition changes to requiring that each of the other $2(\text{ш} - j)$ initially specified blocks contains exactly one error in the information set and exactly one non-S error in the redundancy set. Each of the $i - 2\text{ш}$ remaining blocks contains exactly two non-S errors in the redundancy set. If this error pattern is a codeword then it will be found by the algorithm. Various computer searches did not find parameters leading to smaller operation counts than before even though this approach can produce a larger N.

The literature for low-weight information-set decoding also contains many improvements to the cost of linear algebra. Adapting these improvements to the 2-regular context might be helpful for small ш, but linear algebra becomes negligible as ш grows, so we have not incorporated these improvements into our algorithm. We have also not tried to adapt ball-collision decoding [6] to the 2-regular context.

5 Applications to hash functions

This section gives examples of the cost of finding collisions in several different syndrome-based compression functions: the FSB_{48}, FSB_{160}, and FSB_{256} proposals from [1], and the RFSB-509 proposal from [7]. We repeat a caveat from

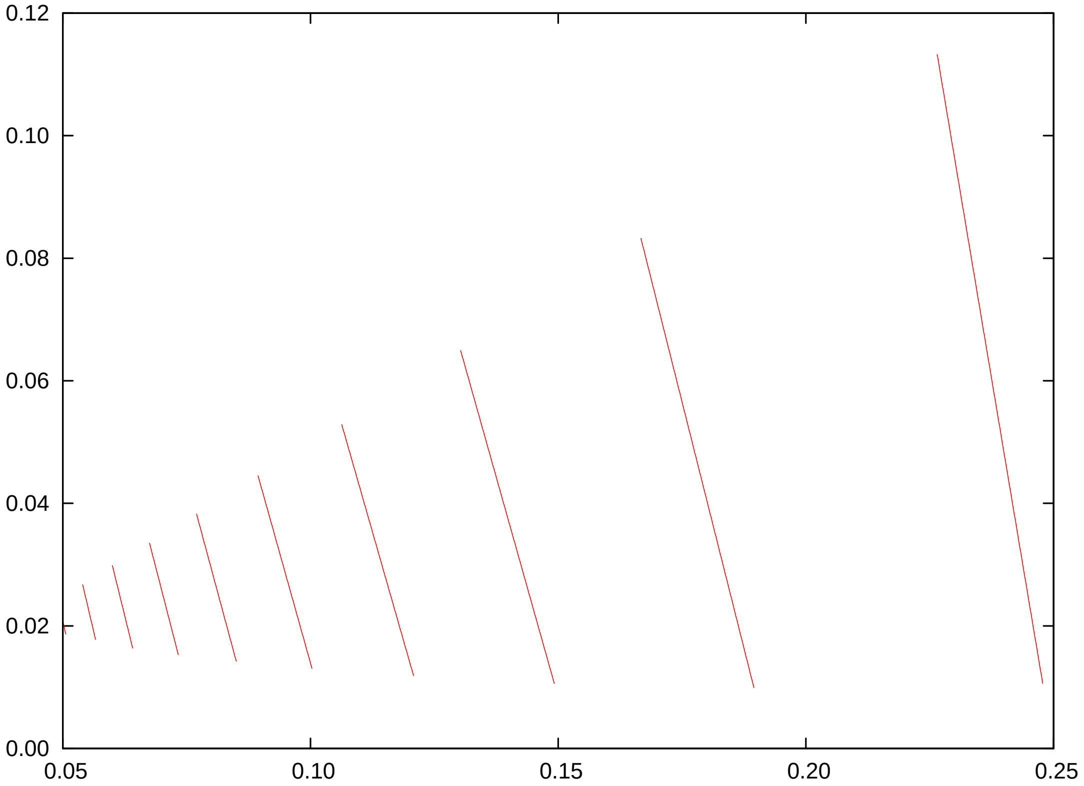

Fig. 4.3. Ш as a function of W_0. See text for description.

Section 1: like previous papers, this paper merely counts the number of bit operations used for arithmetic; we do not claim that our algorithm is an improvement over previous algorithms in models of computation that penalize memory access.

FSB$_{48}$ was designed for $\geq 2^{24}$ collision resistance. It has $r = 192$, $B = 2^{14}$, and $w = 24$. The Augot–Finiasz–Sendrier algorithm needs $2^{75.41}$ iterations, with each iteration using quite a few bit operations. Our algorithm uses just $2^{66.31}$ bit operations with $w_0 = 24$, $\text{ш} = 3$, $N \approx 2^{49.78}$, and $\ell = 48$.

FSB$_{160}$ was designed for $\geq 2^{80}$ collision resistance. It has $r = 640$, $B = 2^{14}$, and $w = 80$. The Augot–Finiasz–Sendrier algorithm needs $2^{251.36}$ iterations. Our algorithm uses just $2^{196.70}$ bit operations with $w_0 = 76$, $\text{ш} = 11$, $N \approx 2^{182.09}$, and $\ell = 184$.

Augot, Finiasz, Gaborit, Manuel, and Sendrier in [1, Table 4, "best attacks known"] claim a "complexity" of "$2^{100.3}$" for "ISD collision search" against FSB$_{160}$. In fact, no attacks are known that find FSB$_{160}$ collisions (equivalently, 2-regular codewords) at this speed. The text in [1, Section 2.2.2] makes clear that [1] is merely estimating the cost of finding a weight-160 codeword, not the cost of finding an 80-block 2-regular codeword.

FSB$_{256}$ was designed for $\geq 2^{128}$ collision resistance. It has $r = 1024$, $B = 2^{14}$, and $w = 128$. The Augot–Finiasz–Sendrier algorithm needs $2^{402.18}$ iterations. Our algorithm uses just $2^{307.56}$ bit operations with $w_0 = 122$, $\text{ш} = 17$, $N \approx 2^{286.92}$, and $\ell = 292$.

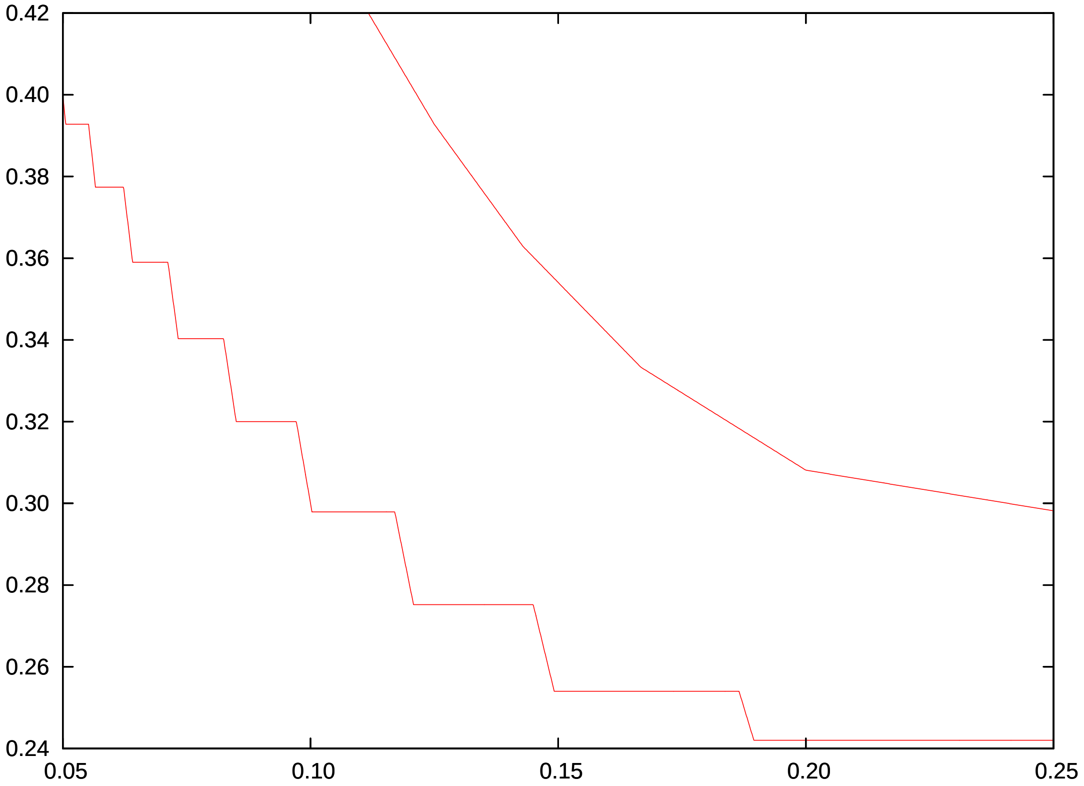

Fig. 4.4. Bottom segments: Asymptotic exponent T for this paper's algorithm (with the restrictions that w_0 divides $r-\ell$ and that $\delta \approx 1$), as a function of W. Top segments: Asymptotic exponent for the Augot–Finiasz–Sendrier algorithm.

RFSB-509 was also designed for $\geq 2^{128}$ collision resistance, but for speed it uses tighter parameters (and a different matrix structure). It has $r = 509$, $B = 2^8$, and $w = 112$. The Augot–Finiasz–Sendrier algorithm needs $2^{154.80}$ iterations, and the Augot–Finiasz–Sendrier analysis would have claimed $2^{152.99}$ iterations. Our algorithm uses just $2^{144.90}$ bit operations with $w_0 = 94$, $\text{ш} = 12$, $N \approx 2^{130.75}$, and $\ell = 133$.

References

[1] Daniel Augot, Matthieu Finiasz, Philippe Gaborit, Stéphane Manuel, Nicolas Sendrier, *SHA-3 proposal: FSB* (2008). URL: http://www-rocq.inria.fr/secret/CBCrypto/fsbdoc.pdf. Citations in this document: §1, §1, §1, §5, §5, §5, §5.

[2] Daniel Augot, Matthieu Finiasz, Nicolas Sendrier, *A fast provably secure cryptographic hash function* (2003). URL: http://eprint.iacr.org/2003/230. Citations in this document: §1, §1, §1, §2, §3, §3, §3, §3, §3, §3, §4.

[3] Daniel Augot, Matthieu Finiasz, Nicolas Sendrier, *A family of fast syndrome based cryptographic hash functions*, in Mycrypt 2005 [13] (2005), 64–83. Citations in this document: §1, §1, §3.

[4] Daniel J. Bernstein, Tanja Lange, Ruben Niederhagen, Christiane Peters, Peter Schwabe, *FSBday: implementing Wagner's generalized birthday attack against the SHA–3 round–1 candidate FSB*, in Indocrypt 2009 [23] (2009), 18–38. URL: http://eprint.iacr.org/2009/292. Citations in this document: §1.

[5] Daniel J. Bernstein, Tanja Lange, Christiane Peters, *Attacking and defending the McEliece cryptosystem*, in PQCrypto 2008 [9] (2008), 31–46. URL: http://eprint.iacr.org/2008/318. Citations in this document: §2, §4, §4.

[6] Daniel J. Bernstein, Tanja Lange, Christiane Peters, *Ball-collision decoding* (2010). URL: http://eprint.iacr.org/2010/585. Citations in this document: §2, §4.

[7] Daniel J. Bernstein, Tanja Lange, Christiane Peters, Peter Schwabe, *Really fast syndrome-based hashing* (2011). URL: http://eprint.iacr.org/2011/074. Citations in this document: §1, §1, §5.

[8] Daniel J. Bernstein, Tanja Lange, Christiane Peters, Henk van Tilborg, *Explicit bounds for generic decoding algorithms for code-based cryptography*, in WCC 2009 (2009), 168–180. Citations in this document: §4.

[9] Johannes Buchmann, Jintai Ding (editors), *Post-quantum cryptography, second international workshop, PQCrypto 2008, Cincinnati, OH, USA, October 17-19, 2008, proceedings*, Lecture Notes in Computer Science, 5299, Springer, 2008. See [5], [14].

[10] Anne Canteaut, Florent Chabaud, *A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511*, IEEE Transactions on Information Theory **44** (1998), 367–378. MR 98m:94043. URL: ftp://ftp.inria.fr/INRIA/tech-reports/RR/RR-2685.ps.gz. Citations in this document: §4.

[11] Aydano B. Carleial, Martin E. Hellman, *A note on Wyner's wiretap channel*, IEEE Transactions on Information Theory **23** (1977), 387–390. ISSN 0018-9448. Citations in this document: §4.

[12] Gérard D. Cohen, Jacques Wolfmann (editors), *Coding theory and applications*, Lecture Notes in Computer Science, 388, Springer, 1989. See [24].

[13] Ed Dawson, Serge Vaudenay (editors), *Mycrypt 2005*, Lecture Notes in Computer Science, 3715, Springer, 2005. See [3].

[14] Matthieu Finiasz, *Syndrome based collision resistant hashing*, in PQCrypto 2008 [9] (2008), 137–147. Citations in this document: §1.

[15] Matthieu Finiasz, Philippe Gaborit, Nicolas Sendrier, *Improved fast syndrome based cryptographic hash functions*, in Proceedings of ECRYPT Hash Workshop 2007 (2007). URL: http://www-roc.inria.fr/secret/Matthieu.Finiasz/research/2007/finiasz-gaborit-sendrier-ecrypt-hash-workshop07.pdf. Citations in this document: §1, §1.

[16] Matthieu Finiasz, Nicolas Sendrier, *Security bounds for the design of code-based cryptosystems*, in Asiacrypt 2009 [20] (2009). URL: http://eprint.iacr.org/2009/414. Citations in this document: §2, §4.

[17] Christoph G. Günther, *Advances in cryptology — EUROCRYPT '88, proceedings of the workshop on the theory and application of cryptographic techniques held in Davos, May 25–27, 1988*, Lecture Notes in Computer Science, 330, Springer-Verlag, Berlin, 1988. ISBN 3-540-50251-3. MR 90a:94002. See [18].

[18] Pil Joong Lee, Ernest F. Brickell, *An observation on the security of McEliece's public-key cryptosystem*, in Eurocrypt '88 [17] (1988), 275–280. MR 0994669. URL: http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/E88/275.PDF. Citations in this document: §2, §4.

[19] Jeffrey S. Leon, *A probabilistic algorithm for computing minimum weights of large error-correcting codes*, IEEE Transactions on Information Theory **34** (1988), 1354–1359. MR 89k:94072. Citations in this document: §2.

[20] Mitsuru Matsui (editor), *Advances in cryptology — ASIACRYPT 2009, 15th international conference on the theory and application of cryptology and information security, Tokyo, Japan, December 6–10, 2009, proceedings*, Lecture Notes in Computer Science, 5912, Springer, 2009. ISBN 978-3-642-10365-0. See [16].

[21] Robert J. McEliece, *A public-key cryptosystem based on algebraic coding theory*, JPL DSN Progress Report (1978), 114–116. URL: `http://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF`. Citations in this document: §2.

[22] Eugene Prange, *The use of information sets in decoding cyclic codes*, IRE Transactions on Information Theory **IT-8** (1962), S5–S9. Citations in this document: §2.

[23] Bimal K. Roy, Nicolas Sendrier (editors), *Progress in Cryptology — INDOCRYPT 2009, 10th international conference on cryptology in India, New Delhi, India, December 13–16, 2009, proceedings*, Lecture Notes in Computer Science, 5922, Springer, 2009. ISBN 978-3-642-10627-9. See [4].

[24] Jacques Stern, *A method for finding codewords of small weight*, in [12] (1989), 106–113. Citations in this document: §2, §4.