

Recovering SNOVA Secret Keys from Biased Vinegar Sampling^{*}

Ward Beullens and Basil Hess

IBM Research, Rüschlikon, Switzerland

Abstract. The Round-3 SNOVA signer samples vinegar variables, which are elements of $\mathbb{F}_q$, by reducing uniform byte strings modulo a power of q . We show that the resulting bias leaks secret-key information for the six odd-characteristic alternative parameter sets. Even though the leakage is small (the most leaky variables leak at most 0.086 bits of information), this still leads to efficient key-recovery attacks which we demonstrate in practice. Key recovery becomes a q -ary LPN problem of dimension ol with a known, full-support error distribution. A naive algorithm needs some ten thousand signatures and 2^{90} to 2^{130} operations, which is already far below the 170 to 331 bits claimed for the six affected sets. Using more signatures makes the attack practical: using standard LPN machinery — BKW reduction with Fourier hypothesis testing — we recover the secret key for four parameter sets in practice using between 9 and 180 million signatures, and at most 16 minutes of wall clock time. Fixing SNOVA to sample the vinegar variables uniformly would prevent the attack completely.

Keywords: Multivariate signatures · SNOVA · UOV · LPN · BKW

1 Introduction

SNOVA is an unbalanced oil-and-vinegar (UOV) signature scheme [8] submitted to the NIST call for additional post-quantum signatures. It obtains compact keys by giving the UOV construction additional algebraic structure over a small field $\mathbb{F}_q$.

Signing requires vectors of randomly chosen vinegar values over $\mathbb{F}_q$. The Round-3 specification [11] produces them by sampling b uniform bytes and reducing the resulting integer modulo q^g . When q is odd, q^g never divides 256^b , so the output is *not* uniform. Some outputs are twice as likely as other outputs. This bias affects the six odd-characteristic alternative parameter sets in Table 6 of the Round-3 specification. (The recommended sets in Table 5 use $q = 16$ and are unaffected by this bias.) Each published signature from an affected set gives

^{*} *AI-assisted research.* This attack, its implementation and the accompanying cost analysis were produced with substantial assistance from AI systems, which proposed and derived key parts of the attack and generated the experimental code. The human authors take full responsibility for the contents of the paper.

noisy linear equations in the rows of the secret transformation, where the noise is a single biased vinegar value. Under Heuristic 1, recovering the rows of the secret transformation is a q -ary LPN problem, and standard LPN machinery solves it. All six affected sets fall far below their claimed security levels, and for four of them we implemented the attack and recovered the key from signatures produced by the unmodified reference code.

The flaw itself is easy to repair, just by sampling the vinegar variables uniformly. For example, the QR-UOV submission [6] uses rejection sampling to ensure that the vinegar variables are uniform, and therefore QR-UOV is not affected by our attack. That vinegar variables are a sensitive target is known from fault attacks [9,7] and from single-trace side-channel analysis [2]. Still, we think the attack is somewhat interesting, because it demonstrates how even a small amount of leakage on the vinegar variables can lead to complete key recovery, re-asserting the fact that implementers need to be very careful to protect the vinegar variables from any kind of side channel leakage or fault injection.

2 SNOVA Signing and Vinegar Sampling

All we need to know about SNOVA to understand our attack is the final step of the signing process, and how the vinegar variables are sampled.

The SNOVA signing process. Signing samples a vinegar matrix $V \in \mathbb{F}_{q^\ell}^{v \times r}$ and somehow obtains a matrix $X \in \mathbb{F}_{q^\ell}^{o \times r}$. The signature is then the pair (Y, X) with

$$Y = V - TX, \quad T \in \mathbb{F}_{q^\ell}^{v \times o}, \quad (1)$$

where T is the secret transformation, called T^{12} in the specification. Throughout this paper, key recovery means recovering T , which together with the public key gives signing capability for arbitrary messages.

Heuristic 1. We model a signature as $(Y, X) = (V - TX, X)$, where V is sampled according to the sampler described below, and X is uniformly random and independent of V .

This model differs from the real signer in two respects: Firstly, X is in bijection with the target hash h_{msg} . This hash is sampled using the same biased sampler as V , so X is not exactly uniform. Secondly, in SNOVA there is a small probability that a signing attempt fails, and the signer retries with a new V and X . This rejection sampling biases the output of the signatures. Nevertheless, our experiments show that we can use Heuristic 1 to simplify the analysis of our attack and still get accurate results.

The (biased) vinegar sampler. For the odd-characteristic alternative parameter sets, the sampler in the SNOVA specification maps b uniform bytes to g elements from $\mathbb{F}_q$, where g and b are chosen such that $q^g \lesssim 256^b$, to minimize the number of random bytes that need to be sampled.

Write $R = 256^b$. The parameters in Table 2 satisfy $q^g \leq R < 2q^g$, so each residue modulo q^g has one or two preimages in $[0, R)$. The sampler draws an integer uniformly from $[0, R)$, reduces it modulo q^g , and outputs the g base- q digits of the result. Identifying $z \in \mathbb{F}_q^g$ with the corresponding integer in $[0, q^g)$, and letting $r_0 = R - q^g$ count the residues with two preimages,

$$\Pr[z] = \begin{cases} 2/R, & z < r_0, \\ 1/R, & z \geq r_0. \end{cases} \quad (2)$$

A fraction r_0/q^g of the outputs is thus drawn exactly twice as often as the rest. Table 1 lists the Kullback–Leibler divergence of the sampled distribution from the uniform one for each field size SNOVA uses. Under Heuristic 1, this divergence bounds the information about T conveyed by the signature coordinates corresponding to one sampler batch, conditional on X . Thus each batch of g vinegar variables leaks at most a tiny fraction of a bit, yet enough signatures allow key recovery.

Table 1. Kullback–Leibler divergence in bits between the sampled distribution on $\mathbb{F}_q^g$ and the uniform one, for each field size q used by SNOVA.

q	KL divergence
11	0.0755
13	0.0859
16	0.0000
19	0.0730

Which entries of V are biased most. The sampler outputs elements of $\mathbb{F}_q$ in batches of size g . These g elements are not identically distributed. Let z be the number sampled uniformly at random in $[0, R)$. Then the first element in the batch is simply $z \bmod q$, which is close to uniform. In contrast, the last digit is $\lfloor z/q^{g-1} \rfloor \bmod q$, which is biased much more strongly. Let the number $r_0 = R - q^g$ of doubly covered residues be $r_0 = a q^{g-1} + s$ with $0 \leq s < q^{g-1}$. Then the distribution of the last digit d of each batch returned by the sampler is equal to:

$$\Pr[d] = \begin{cases} 2q^{g-1}/R, & d < a, \\ (q^{g-1} + s)/R, & d = a, \\ q^{g-1}/R, & d > a. \end{cases} \quad (3)$$

Call $S = \{0, \dots, a-1\}$ the *heavy set*: each of its a values is drawn exactly twice as often as each of the values above a . The digit a itself is drawn with some probability that is between that of the digits in S and those above a . The most biased digit lands in S with probability

$$P_1 = \frac{2a q^{g-1}}{R}, \quad (4)$$

whereas a uniform digit lands in S with probability $P_0 = a/q$. The gap $\delta = P_1 - P_0$ lies between 0.13 and 0.17 for every odd field SNOVA uses. Table 2 lists the value of the gap for the fields used by SNOVA.

Table 2. The heavy set $S = \{0, \dots, a-1\}$, for each odd field size SNOVA uses. A biased digit of V lands in S with probability P_1 ; a uniformly random one lands in S with probability $P_0 = a/q$.

q	(g, b)	a	P_0	P_1	$\delta = P_1 - P_0$
11	(16, 7)	6	0.5455	0.6957	0.1502
13	(15, 7)	5	0.3846	0.5464	0.1618
19	(15, 8)	4	0.2105	0.3465	0.1360

The sampler fills the coefficients of the matrix $V \in \mathbb{F}_{q^\ell}^{v \times r}$ in a fixed public order (specified in Algorithm 14 in the SNOVA specification). Specifically, the rows of V are filled consecutively, and row V_i takes the ℓr consecutive stream elements $[i\ell r, (i+1)\ell r)$. The most biased digits occupy every g -th position of the stream, so V_i contains at least $\lfloor \ell r/g \rfloor$ biased field elements. Tables 2 and 3 show that $\ell r \geq g$ for every affected set, so this number is at least 1.

For each row V_u of V we can therefore fix a position (α_u, β_u) of V_u that receives a biased digit, β_u selecting an entry of V_u and α_u selecting an $\mathbb{F}_q$ -coefficient within the β_u -th $\mathbb{F}_{q^\ell}$ -element. Since ℓr need not be a multiple of g , that position depends on u .

3 A Naive Key-Recovery Algorithm

In this section we give a simple algorithm to recover the secret key given a set of SNOVA signatures. The algorithm recovers the secret key T row by row. For each row index $u \in \{0, \dots, v-1\}$, we iterate over all the $q^{o\ell}$ possible values this row could take. Then, for each guess, we can recover the vinegar values in row u , using $V = Y + TX$, where Y and X are part of the signature. If the guess is correct, then the row of V will look like it is sampled using the biased sampler. If the guess was wrong, then the recovered row of V is uniformly random under Heuristic 1. We can distinguish the correct guess by simply counting the number of times that a biased digit in the row of V is in the heavy set S .

This approach is easy to analyse and does not need many signatures, but has the drawback that it is not very efficient, since we need to iterate over all the $q^{o\ell}$ possible values of the rows, so we cannot perform it in practice for the affected SNOVA parameters, even though its cost is far below the NIST target security levels. Section 4 describes a much faster algorithm that we implemented and ran, however, this algorithm requires more signatures.

3.1 Reconstructing vinegar values from a guess

Fix a row index $u \leq v$ and write T_u for row u of T , and V_u, Y_u for the corresponding rows of V and Y . Row u of Equation (1) involves no other row of T , so

$$V_u = Y_u + T_u X, \quad (5)$$

and a guess for T_u turns every signature (Y, X) into a reconstruction of that signature's vinegar values in row u . The row T_u consists of o elements of $\mathbb{F}_{q^\ell}$, each of which is ℓ coefficients over $\mathbb{F}_q$, so a guess is an element of $\mathbb{F}_q^{o\ell}$ and there are $q^{o\ell}$ of them to consider.

Of the ℓr coefficients of V_u only one interests us: the one at the position (α_u, β_u) that Section 2 singles out as a biased digit of a sampler batch. (Only for parameter set E is there guaranteed to be two biased digits in every row of V .) Write $e_i(t)$ for that coefficient as reconstructed from the i -th signature under a guess $t \in \mathbb{F}_q^{o\ell}$. Equation (5) is $\mathbb{F}_q$ -linear in the coefficients of the guess, so

$$e_i(t) = y_i + \langle x_i, t \rangle \quad (6)$$

for a scalar $y_i \in \mathbb{F}_q$ and a vector $x_i \in \mathbb{F}_q^{o\ell}$, both of which are publicly computable from the signature. For the correct guess of $t = T_u$, the reconstructed $\mathbb{F}_q$ element comes from the sampler, so we have

$$e_i(T_u) \sim p, \quad (7)$$

where p is the distribution (3) of a last sampler digit. For an incorrect guess, $e_i(t)$ is uniform over $\mathbb{F}_q$ under Heuristic 1: the nonzero difference $t - T_u$ introduces a uniform offset $\langle x_i, t - T_u \rangle$.

3.2 Counting heavy digits

Given N signatures and a guess $t \in \mathbb{F}_q^{o\ell}$, count how many of the reconstructions land in the heavy set S of Section 2:

$$c(t) = \#\{i \leq N : e_i(t) \in S\}. \quad (8)$$

Then, $c(T_u)$ is binomial with parameters N and P_1 , while for a wrong guess $c(t)$ is binomial with parameters N and $P_0 < P_1$. Accepting a guess when its count exceeds the midpoint threshold

$$\tau = N \frac{P_0 + P_1}{2} \quad (9)$$

therefore singles out the correct row. The whole attack is: Collect a set of N signatures. Then, for each row index $u \in \{1, \dots, v\}$, and for every $t \in \mathbb{F}_q^{o\ell}$, compute the count $c(t)$ of Equation (8). With high probability, for each row index u there is exactly one row candidate t whose count $c(t)$ exceeds the threshold τ . Output the secret key T as the matrix consisting of those rows.

3.3 How many signatures are needed

Under Heuristic 1, we compute a sufficient number of signatures N for the attack to succeed with high probability. That means we need for all the wrong $t \neq T_u$ that $c(t) < \tau$ and for the correct $t = T_u$ we need $c(T_u) > \tau$. Hoeffding's inequality bounds both kinds of failure by

$$\Pr[\text{Bin}(N, P_0) \geq \tau] \leq e^{-N\delta^2/2}, \quad \Pr[\text{Bin}(N, P_1) < \tau] \leq e^{-N\delta^2/2}, \quad (10)$$

since τ sits $\delta/2$ away from either mean. A union bound over the $q^{o\ell} - 1$ wrong guesses and the correct one puts the chance of getting a single row wrong at $q^{o\ell}e^{-N\delta^2/2}$. We ask for that to be at most $2^{-32}/v$, so that a second union bound, this time over the v rows, leaves the whole of T correct with probability at least $1 - 2^{-32}$.

$$N = \left\lceil \frac{2(o\ell \ln q + \ln v + 32 \ln 2)}{\delta^2} \right\rceil. \quad (11)$$

The method with the threshold τ is very crude, but we chose it for its simplicity.

Cost of the attack. The bottleneck is computing $c(t)$ for all candidates, which costs $o\ell$ multiply-adds per sample and candidate, so the enumeration costs $N o\ell q^{o\ell}$ per row; Extracting the LPN samples from the signatures is negligible in comparison. Over all v rows,

$$T_{\text{direct}} = v N o\ell q^{o\ell}. \quad (12)$$

(Enumerating the candidates in Gray-code order would remove the factor $o\ell$; we do not bother, since the next section replaces the enumeration altogether.) Table 3 collects the resulting figures.

Table 3. The six affected alternative parameter sets [11, Table 6], the number N of signatures the naive algorithm needs, and its cost. The biased-elements column gives the average over sampler alignments, $\ell r/g$. An operation is one small-field multiply-add or something of comparable cost; it is not a Boolean gate, so $\log_2 T_{\text{direct}}$ is not directly comparable with the last column, which gives the security claimed for the set in classical gates.

Set	(v, o, q, ℓ, r)	Biased elements per row of V	N	$\log_2 T_{\text{direct}}$	Claimed
		$(\ell r/g)$			
A	(27, 4, 19, 4, 6)	8/5	7851	89.66	201
B	(27, 5, 13, 4, 4)	16/15	5865	95.60	170
C	(39, 5, 11, 4, 6)	3/2	6543	91.47	230
D	(38, 5, 19, 4, 6)	8/5	9162	107.69	269
E	(40, 5, 19, 5, 6)	2	10759	129.56	331
F	(47, 6, 19, 4, 6)	8/5	10458	125.44	318

We conclude that with this version of the attack, the number of signatures is relatively small: under Heuristic 1, eleven thousand signatures on arbitrary messages suffice for every affected parameter set, and the cost of the attack is far below the claimed security levels. However, it does not seem practical to demonstrate a key recovery in practice with this approach.

4 The Practical Attack

It should be clear by now that the $e_i = \langle x_i, T_u \rangle + y_i$ equations are q -ary LPN samples with a known, full-support error distribution $e_i \sim p$. The algorithm from the previous section is a naive approach to solve this LPN problem, which requires few samples but a long runtime. In this section we apply some standard LPN machinery to solve the problem more efficiently, at the cost of requiring more samples. We combine BKW-style collision reduction [3,1] with a Fourier decoder [10,5].

4.1 BKW reduction

The bottleneck of the naive algorithm is the enumeration of all $q^{o\ell}$ candidates for a row of T . If we had sufficiently many LPN samples $\{(x_i, y_i)\}_i$, where all but the last d entries of the x_i vectors are zero, then we could enumerate just the last k coordinates of T_u , to find the last d entries of T_u with only q^d work. Once part of T_u is found, we can iterate the process to find the remaining part. The BKW reduction algorithm is a way to produce such samples by iteratively subtracting pairs of LPN samples that collide in a block of coordinates.

We start from a large set of N samples $\{(x_i, y_i)\}_i$. To eliminate a block of d_1 coordinates, bucket samples by the first d_1 entries of the x_i vectors. If (x, y) and (x', y') are two samples in the same bucket, their difference satisfies

$$e - e' = \langle x - x', T_u \rangle + (y - y'), \quad (13)$$

and the chosen block of $x - x'$ is zero. The distribution of the new error term $e - e'$ is the convolution of the original distribution with its reflection around 0, which decreases the bias. After k iterations of this process with block sizes $d_1, \dots, d_k$, we have eliminated $\sum_{j=1}^k d_j$ coordinates from the vectors, at the cost of convolving the error distribution with itself k times. Starting with $M_0 = N$ samples, we can estimate the number of samples we have after i iterations of the BKW step with the recurrence relation

$$M_i \simeq \frac{\binom{M_{i-1}}{2}}{q^{d_i}}. \quad (14)$$

The useful bias after reduction is most conveniently described in the Fourier domain. Put $\zeta_q = \exp(2\pi i/q)$ and let

$$\phi_h = \mathbb{E}_{e \sim p} [\zeta_q^{he}] = \sum_{z \in \mathbb{F}_q} p(z) \zeta_q^{hz}. \quad (15)$$

The error $e - e'$ after one cancellation has Fourier coefficients $|\phi_h|^2$. A second cancellation subtracts two such differences and gives $|\phi_h|^4$, and so on. The dimension reduction therefore squares the bias at each level, explaining why, compared to the naive attack, many more samples are required at the final stage.

Remark (reusing collision matching). The selected positions can coincide for different rows. If $(\alpha_u, \beta_u) = (\alpha_{u'}, \beta_{u'})$, then the two LPN instances have the same vector x_i for every signature i , although their right-hand sides and secret rows differ. The collision buckets and the selected pairs at both reduction levels depend only on the vectors x_i . Their matching can therefore be computed once and reused for all rows with the same selected position; only the right-hand-side arithmetic and Fourier decoding remain row-specific.

4.2 Fourier hypothesis testing

Let $(z_i, c_i)_{i=1}^{M_k}$ be the samples after the k BKW reductions, with z_i of length $d = o\ell - \sum_{j=1}^k d_j$. We want to find the correct candidate $w \in \mathbb{F}_q^d$ for the last d elements of the row T_u .

We focus on the first nontrivial Fourier coefficients of the error distributions, because they capture the most bias. Let ψ_i be the first nontrivial Fourier of the error distribution after i BKW reductions, so $\psi_0 = \phi_1$, and $\psi_i = |\phi_1|^{2^i}$, as described above. Table 4 shows how quickly this coefficient decreases under repeated BKW reductions.

Table 4. The first Fourier coefficient and its successive powers for the convoluted error distributions.

q	$\psi_0 = \phi_1$	$\psi_1 = \phi_1 ^2$	$\psi_2 = \phi_1 ^4$	$\psi_3 = \phi_1 ^8$
11	$0.015 + 0.198i$	$3.92 \cdot 10^{-2}$	$1.53 \cdot 10^{-3}$	$2.37 \cdot 10^{-6}$
13	$0.109 + 0.187i$	$4.66 \cdot 10^{-2}$	$2.18 \cdot 10^{-3}$	$4.75 \cdot 10^{-6}$
19	$0.143 + 0.081i$	$2.69 \cdot 10^{-2}$	$7.27 \cdot 10^{-4}$	$5.28 \cdot 10^{-7}$

The Fourier hypothesis testing of [10,5] uses the scoring function

$$D_k(w) = \operatorname{Re} \left(\frac{\overline{\psi_k}}{|\psi_k|} \left(\sum_{i=1}^{M_k} \zeta_q^{c_i + \langle z_i, w \rangle} \right) \right). \quad (16)$$

This is the projection of the (unnormalized) sample estimate of the first nontrivial Fourier coefficient of the residual error distribution (the quantity in the inner bracket) onto the direction of the expected value ψ_k of that coefficient. Hence the expected score for the correct candidate w is $M_k |\psi_k|$, while for a wrong candidate, under Heuristic 1, the expected score is zero, since the residual error distribution is uniform. Either score has variance approximately $M_k/2$, so their difference has variance approximately M_k . Therefore, if the number of samples

M_k is large enough, the correct candidate will have a significantly higher score than any wrong candidate. So the algorithm computes all the scores and outputs the candidate with the highest score as the correct one. Moreover, this scoring function is used because it can be computed for all candidates w simultaneously using a single d -dimensional Fourier transform. (See [5] for more details.)

Once the last d entries of the row T_u are found, we can back-substitute the solution into the LPN samples of the previous BKW stage and recover the next block of T_u with the scoring function D_{k-1} . This is iterated until all of T_u is found, and repeated for each row T_u of T .

4.3 Heuristic cost analysis

We now estimate the number of signatures and the work required for the attack. We start by estimating the number of samples required for the Fourier hypothesis testing to pick the correct candidate with high probability.

Since the scoring function is a sum over many (mostly) independent samples, we use the central limit theorem to model the difference between the score of the correct candidate and that of a wrong candidate as a Gaussian random variable with mean $M_k|\psi_k|$ and variance M_k . A fixed wrong candidate then wins with probability approximately $Q(\sqrt{M_k}|\psi_k|)$, where Q is the upper tail of a standard normal variable. A union bound over the $q^d - 1$ wrong candidates in each of the v rows gives the criterion

$$v(q^d - 1)Q(\sqrt{M_k}|\psi_k|) \leq 0.1, \quad (17)$$

for choosing the required number of samples at the last BKW stage M_k . This targets a failure probability of at most 10% across all rows. We then invert the recurrence (14), to obtain an estimate for the required number of signatures $N = M_0$. This remains a heuristic estimate because dependencies between derived equations are intrinsic to all-pairs collision reduction [4]. We estimate the work factor as

$$\#ops = v \left(\sum_{j=1}^k M_j + \sum_{j=1}^k q^{d_j} \log_2(q^{d_j}) \right), \quad (18)$$

counting one unit of work per derived equation in the BKW step and one unit per complex FFT operation in the Fourier step.

To estimate the memory requirement, let b_j be the packed byte width of one level- j equation, including its right-hand side and the origin identifiers needed to exclude repeated raw observations. The required memory is then

$$\sum_{j=0}^{k-1} b_j M_j + 8q^{d_k}, \quad (19)$$

using four-byte registers for the FFT. Table 5 reports the resulting estimates for the required number of signatures, the required amount of work and the memory requirements.

Table 5. Estimated signature, work, and memory exponents for the practical attack.

Set	Split ($d_1 + \dots + d_k + d$)	# signatures ($\log_2$)	Work factor ($\log_2$)	Memory
A	5 + 5 + 6	23.23	35.14	536 MiB
B	6 + 7 + 7	24.09	36.42	1002 MiB
C	6 + 7 + 7	23.18	35.22	457 MiB
D	6 + 7 + 7	27.53	40.92	14.83 GiB
E	6 + 6 + 6 + 7	28.09	52.59	2.02 TiB
F	6 + 6 + 6 + 6	29.06	52.65	1.45 TiB

5 Experimental Results

We ran the attack successfully against parameter sets A–D. In each case we recovered the complete secret transformation T , and used this key to produce signatures which are accepted by the reference implementation of the SNOVA verifier. Table 6 gives the measured recovery costs. Our implementation is publicly available at

https://github.com/WardBeullens/SNOVA_LPN_Attack

Table 6. Measured complete key-recovery runs. Here J is the maximum number of concurrent row jobs; the memory-mapped LPN input is shared between them, and peak RAM is the aggregate memory usage.

Set	Signatures	J	Shared LPN input	Wall time	Peak RAM
A	9 million	27	0.90 GiB	21.8 s	13.6 GiB
B	16.5 million	27	5.02 GiB	34.4 s	26.7 GiB
C	9 million	32	0.66 GiB	22.3 s	6.45 GiB
D	180 million	6	23.13 GiB	16 min 11 s	108.5 GiB

The measurements used a 16-core, 32-thread Intel Xeon Gold 6346 at 3.10 GHz. Sets A–C used up to 32 concurrent row jobs. Memory limited set D to six concurrent row jobs. Generating its 180 million full message–signature pairs took 40 hours using the SNOVA reference implementation.

Figure 1 shows the fraction of correctly recovered rows for set B as the number of signatures varies. The transition to complete recovery occurs close to our heuristic estimate, providing experimental evidence that the estimated number of required signatures is reasonably accurate.

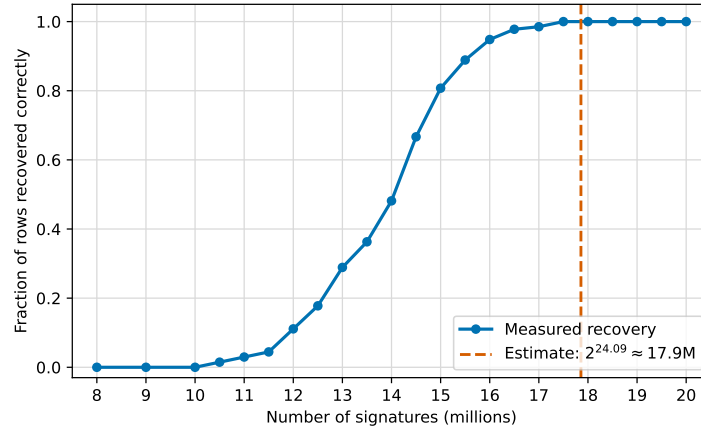


Fig. 1. Fraction of rows recovered correctly for the key-recovery attack against parameter set B, as a function of the number of signatures used. Each data point is the average of five runs of the attack. The dashed line is our heuristic estimate $N = 2^{24.09}$ for the number of signatures needed to complete the attack with high probability, obtained with the methodology of Section 4.3.

References

1. Albrecht, M.R., Cid, C., Faugère, J.C., Fitzpatrick, R., Perret, L.: On the complexity of the BKW algorithm on LWE. DCC **74**(2), 325–354 (2015). <https://doi.org/10.1007/s10623-013-9864-x>
2. Aulbach, T., Campos, F., Krämer, J., Samardjiska, S., Stöttinger, M.: Separating oil and vinegar with a single trace side-channel assisted kipnis-shamir attack on UOV. IACR TCHES **2023**(3), 221–245 (2023). <https://doi.org/10.46586/tches.v2023.i3.221-245>
3. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. In: 32nd ACM STOC. pp. 435–440. ACM Press (May 2000). <https://doi.org/10.1145/335305.335355>
4. Devadas, S., Ren, L., Xiao, H.: On iterative collision search for LPN and subset sum. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part II. LNCS, vol. 10678, pp. 729–746. Springer, Cham (Nov 2017). https://doi.org/10.1007/978-3-319-70503-3_24
5. Duc, A., Tramèr, F., Vaudenay, S.: Better algorithms for LWE and LWR. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 173–202. Springer, Berlin, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46800-5_8
6. Furue, H., Ikematsu, Y., Kiyomura, Y., Takagi, T.: A new variant of unbalanced Oil and Vinegar using quotient ring: QR-UOV. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part IV. LNCS, vol. 13093, pp. 187–217. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92068-5_7
7. Kato, T., Kiyomura, Y., Takagi, T.: Improving fault attacks on rainbow with fixing random vinegar values. In: Cheng, C.M., Akiyama, M. (eds.) IWSEC 22. LNCS,

- vol. 13504, pp. 147–165. Springer, Cham (Aug / Sep 2022). https://doi.org/10.1007/978-3-031-15255-9_8
8. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar signature schemes. In: Stern, J. (ed.) EUROCRYPT’99. LNCS, vol. 1592, pp. 206–222. Springer, Berlin, Heidelberg (May 1999). https://doi.org/10.1007/3-540-48910-X_15
 9. Krämer, J., Loiero, M.: Fault attacks on UOV and Rainbow. In: Polian, I., Stöttinger, M. (eds.) COSADE 2019. LNCS, vol. 11421, pp. 193–214. Springer, Cham (Apr 2019). https://doi.org/10.1007/978-3-030-16350-1_11
 10. Leveil, É., Fouque, P.A.: An improved LPN algorithm. In: De Prisco, R., Yung, M. (eds.) SCN 06. LNCS, vol. 4116, pp. 348–359. Springer, Berlin, Heidelberg (Sep 2006). https://doi.org/10.1007/11832072_24
 11. Wang, L.C., Chou, C.Y., Ding, J., Guo, H., Kuan, Y.L., Leegwater, J.A., Li, M.S., Li, P., Tseng, B.S., Tseng, P.E., Wang, C.C.: SNOVA: Proposal for NIST PQC: Additional digital signature schemes (Aug 2026), https://snova.pqclab.org/files/SNOVA_Round3.pdf, version 3.0