About Wave Implementation and its Leakage Immunity

Thomas Debris-Alazard, Nicolas Sendrier, Jean-Pierre Tillich

▶ **To cite this version:**

Thomas Debris-Alazard, Nicolas Sendrier, Jean-Pierre Tillich. About Wave Implementation and its Leakage Immunity. 2019. hal-02424231

HAL Id: hal-02424231
https://inria.hal.science/hal-02424231

Preprint submitted on 26 Dec 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

About Wave Implementation and its Leakage Immunity*

Thomas Debris-Alazard[1,2], Nicolas Sendrier[2], and Jean-Pierre Tillich[2]

[1] Sorbonne Universités, UPMC Univ Paris 06
[2] Inria, Paris
{thomas.debris,nicolas.sendrier,jean-pierre.tillich}@inria.fr

May 22, 2019

Abstract

Wave is a recent digital signature scheme [3]. It is based on a family of trapdoor one-way Preimage Sampleable Functions and is proven EUF-CMA in the random oracle model under two code-based computational assumptions. One of its key properties is to produce signatures uniformly distributed of fixed Hamming weight. This property implies that, if properly implemented, Wave is immune to leakage attack. We describe here the key stages for the implementation of the Wave trapdoor inverse function to integrate all the features to achieve leakage-freeness. A proof of concept implementation was made in SageMath. It allowed us to check that properly generated Wave signatures are uniformly distributed. In particular, we show that the signatures produced by this implementation defeat the Barreto-Persichetti attack. We show which features of the Wave specification were improperly put aside and explain why the claim of breaking Wave is incorrect.

Preliminary Statement. We consider here the first version of Wave (v1 of [3]). This is the version which is allegedly broken in [1]. All the features that guaranty the absence of leakage are already there. It considers a strict $(U, U+V)$ code (*i.e.* not generalized) and has no information set gap. The description of the decoder for generalized $(U, U+V)$ codes only appeared in the v2. The information set gap (denoted d) appeared in third version of Wave. The information set gap was introduced to give a provably small upper bound for the statistical distance between the signatures distribution and uniform distribution. We conjecture that this statistical distance is negligible even with a zero gap. Our implementation is available at the following URL: `https://project.inria.fr/wave` and includes the three versions.

1 Hash-and-Sign Signatures and Leakage Attacks

A hash-and-sign digital signature scheme uses a trapdoor one-way function $f()$. The message is hashed to produce a random element y in the domain of $f()$. The legitimate user, the signer, uses the trapdoor to compute a preimage x of y. The signature x is verified by checking $f(x) = y$.

When the one-way function $f()$ is surjective, the trapdoor inverse function will use the secret trapdoor to select one particular preimage. It may happen that this selection is biased and leaks information on the secret key.

*This work was supported by the ANR CBCRYPT project, grant ANR-17-CE39-0007 of the French Agence Nationale de la Recherche.

2 Reaching a Target Distribution for Wave Signatures

The Wave signature scheme is hash-and-sign with a surjective one-way function. Moreover, as noticed in the design of Surf[1], the binary ancestor of Wave, the "native" $(U, U+V)$ decoder is highly biased and the signature algorithm has to be carefully designed to avoid leakage attacks. Here the target distribution is the uniform distribution over words a constant weight w. To reach this distribution we proceed with the following stages.

1. Select parameters, code length and dimension (n, k), signature weight w, but also the component codes dimension $k_U + k_V = k$. This choice is made so that the two related computational problems are hard enough and the "native" decoder produces an error of average weight w.

2. The decoder has two successive steps.

   2a. The first decoding step draws an integer $\ell \in [0, k_V]$ according to a distribution $\mathcal{D}_V$, then draws a set of k_V random positions and fills them with random values of Hamming weight ℓ. The word is completed into an output $\mathbf{e}_V$ of Hamming weight t.

   2b. The second decoding step depends of t. It draws an integer $k_{\neq 0} \in [0, k_U]$ according to a distribution $\mathcal{D}_U^t$, then draws a set of k_U random positions, $k_{\neq 0}$ of which are in the support of $\mathbf{e}_V$. The k_U positions are filled and completed as specified (Algorithm 4 and Algorithm 5 in [3]). This is repeated until the final error vector has weight exactly w. Note that this requires a family of distributions, one for each value the Hamming weight t of the first step output.

3. After each decoding step we apply a rejection sampling vector. The rejection vectors are precomputed.

   3a. The result of the first decoding step is accepted with probability $\mathbf{r}_V(t)$, $t = |\mathbf{e}_V|$. There is a single rejection vector.

   3b. The result of the second decoding step is accepted with probability $\mathbf{r}_U(t, \ell)$, $t = |\mathbf{e}_V|$ and ℓ depends of $\mathbf{e}_U$ and $\mathbf{e}_V$. There is a family of rejection vectors, one for each value of t.

   Note that there is some freedom in the choice of the distributions $\mathcal{D}_V$ and $\mathcal{D}_U^t$. Each choice will lead to different rejection vectors. Note also that those distributions must be chosen carefully else the acceptance ratio may become exponentially small. Here we chose truncated Laplace distributions.

This framework and more details on the choice of the distributions and of the rejection vectors are given in [3].

3 Leakage Attacks on Wave

If the three stages of §2 are correctly implemented, signatures are produced according to a uniform distribution over S_w the set of ternary words of length n and weight w.

In the random oracle model, to perform a leakage attack, an adversary will observe any number of signatures. After some extra computation, the adversary tries to extract some information on the secret key.

[1] Surf was later abandoned because one of the computational hardness assumptions was not verified

Full Implementation: Each signature is drawn independently and uniformly in S_w. The sample is indistinguishable from a random set of words of weight w. Obviously no leakage attack is possible.

Removing Stage 3: If for some reason no rejection vector is applied, the adversary will still have a hard time. If the distributions $\mathcal{D}_V$ and $\mathcal{D}_U^t$ have the correct mean values (see [3, §5.3]) the observed distribution will still be close to uniform.

Modifying Stage 2: If the distributions are not correctly chosen, then things can get really bad. First, the rejection vectors can be computed but this will often lead to an exponentially small acceptance rate. Without the rejection vectors, the first order statistics on coordinate pairs (i, j) will reveal some information. We call first order statistic a probability $\mathbb{P}((e_i, e_j) = (a, b))$ for some $(a, b) \in \mathbb{F}_3^2$ where e_i and e_j are the i-th and j-th coordinates of a signature $\mathbf{e}$, here coming from the modified algorithm. In fact, the first order statistics when (i, j) is a matched pair (*i.e.* i and j are symmetric to one another in the non permuted version of the code), are different from those of the other pairs. This fact was remarked in a previous work in the binary case [2, §4.1].

We illustrate the various modifications and their impact on first order statistics in §6.

4 Implementing Wave

As proof a proof of concept, we implemented Wave as described in §2. This was done in SageMath and allowed us to check that the output distribution was not distinguishable from the uniform distribution over S_w. It is available on Wave's web page[2].

Comments and Limitations.

1. As mentioned above, the choice for the distributions in stages 2a and 2b is free but a bad choice could lead to a low acceptance rate in stage 3.

2. If the mean values of the distributions are correct (*i.e.* those that stem from a final output uniformly distributed in S_w) there is no first order bias even without rejection vectors, but to completely avoid leakage, rejection vectors are needed in stages 3a and 3b.

3. The acceptance rate is exponentially small if distributions are not well chosen. It is the case for instance by chosing binomial distributions, even with the correct means. What happens is that the tails of the output distributions are too low and emulating those tails correctly will force a low average acceptance rate. The input distribution is best chosen such that the corresponding output tails augment above those of the target distribution.

4. An open question which we have not fully addressed yet is the precision with which the rejection vectors must be computed to reach a given security level against leakage attacks.

5. The current program uses truncated Laplace distributions and the native floating point arithmetic of SageMath to compute the rejection vectors. Because of the fixed precision arithmetic, the resulting signature algorithm leaks. This leakage is small, but it certainly needs to be quantified precisely. Nevertheless, what we present here is enough to judge of the feasability of a leakage-free implementation.

[2] `https://project.inria.fr/wave`

5 The Wrongful Approach of Barreto and Persichetti

Barreto and Persichetti claim an attack on Wave in [1]. It is a leakage attack but on a scheme deprived of its substance. The signatures are produced with an algorithm where stage 3 and most of stage 2 are removed. It ignores $\mathcal{D}_V$, picks and fills the k_V information positions randomly. It ignores $\mathcal{D}_U^t$ and picks the k_U information positions randomly. Indeed, signatures produced that way are highly biased and the secret key is easily recovered. The statistic they use is $\mathbb{P}(e_i = -e_j) - \mathbb{P}(e_i = e_j)$ (see §6).

We formally contest that this is an attack on Wave. To claim an attack on Wave one has to produce the signatures with either a program compliant with the specification, or, if it is not the case, one has to *prove* that the unimplemented features are irrelevant for the attack. Of course we wrote to the authors. This was greeted by a new preprint version in which a few remarks were added. But the position of the authors was unchanged and expressed by the following paragraph:

> "Summarizing: our attack still holds against a fully detailed implementation of Wave; it does not depend on the total counts of signature coefficient values being uniform, and is not thwarted by making them so."

The beginning of the sentence seems to imply that they are aware that a "fully detailed implementation" was specified, that they did not use those full specifications, and that they believe it does not matter. We are are unable to comment the rest of the paragraph.

There was yet another version of [1]. After claiming in the first version that they "do not see how the scheme can be repaired", Barreto and Persichetti proposed a fix. This fix, called Tsunami, consisted in changing the distribution of the total weight of the information position values in stage 2a, in other words adjusting the distribution $\mathcal{D}_V$. By doing this, the difference of the statistic $\mathbb{P}(e_i = -e_j) - \mathbb{P}(e_i = e_j)$ between the matched pairs and the others vanishes. However, as we show in the next section, Tsunami is still prone to leakage attacks using other first order statistics, for instance $\mathbb{P}((e_i, e_j) = (0, 0))$.

After we completed the full implementation of Wave with all the design stages, we checked the attack, which does not work anymore. We sent them a sample of 100 000 signatures and a public key on February 19. The paper is still online with the following claim:

> "The number of legitimate signatures the attacker needs to gather is fairly small (around 600 for the proposed 128-bit secure Wave parameters). The equivalent key recovery runs very fast in practice (a few seconds). The most time-consuming stage by far is the generation of the collected legitimate signatures (about one minute)."

Thanks to our implementation, this claim is verifiably false if the attack is applied to genuine Wave signatures. The measured statistics of §6 are consistent with the theoretical results of [3] which prove that neither the statistic used in [1], nor any other first order statistic, nor any statistic at all, can succeed in distinguishing the matched pairs of coordinates from the others.

6 Measured Statistics

We used our implementation to produce Wave signatures. We also extracted the Magma script from [1] to produce signatures on which the attack works. The Magma script of [1] do not follow all specifications of Wave but the authors claim that their attack would work even on a fully detailed implementation. We show below that it is not the case.

The tested 4 categories of signatures. First genuine Wave signatures. Next Wave signatures in which the stage 3 is not applied. The last two were produced with the Magma script of

[1]. The "raw decoder" first, we refer here to the algorithm on which the attack applies. Then signatures using the Tsunami fix.

For each of those categories, we generated a set E of signatures (all of them with the same key) and checked the first order statistics that is, for each $(a, b) \in \mathbb{F}_3^2$, each $\mathbf{e} = (e_1, \ldots, e_n) \in E$, we counted for each pair of positions (i, j) the proportion of signatures such that $(e_i, e_j) = (a, b)$.

For a random $\mathbf{e}$ of Hamming weight w we have for all pairs of position (i, j) and all pair of values (a, b) (of weight $\delta \in \{0, 1, 2\}$)

$$\pi_\delta = \mathbb{P}((e_i, e_j) = (a, b)) = \frac{\binom{n-2}{w-\delta}}{\binom{n}{w} 2^\delta}$$

When the set E is generated with a bias, it may happen that some pairs have different values. As mentionned in [2, §4.1], the "native" $(U, U + V)$ decoder is biased and some of the above statistic differ from their expectation for matched pairs. A matched pair of position has the form $(\ell, \ell + n/2)$ in the non permuted version of the code (the secret).

We chose the parameters $n = 5172$ and $w = 4980$ from the first version of Wave. This is good enough to provide evidence that the Wave signatures are correctly distributed and allows a comparison with [1]. For those parameters, we have $\pi_0 = 0.0013712$, $\pi_1 = 0.017876$, and $\pi_2 = 0.231781$.

All the statistics produced below can be easily reproduced from publicly available software.

6.1 Wave Signatures

Below the percentage for each value of (a, b) on average over all pairs and on average over all matched pairs. Those values were obtained from a sample of 400 000 signatures produced by our SageMath full implementation of Wave. Note that there are $n(n-1)/2$ distinct pairs and only

Table 1: Wave

(a, b)	$(0, 0)$	$(1, 0)$	$(2, 0)$	$(0, 1)$	$(0, 2)$	$(1, 1)$	$(2, 1)$	$(1, 2)$	$(2, 2)$
matched	0.13710	1.7876	1.7876	1.7878	1.7874	23.1780	23.1797	23.1768	23.1780
all	0.13712	1.7875	1.7875	1.7877	1.7877	23.1783	23.1782	23.1780	23.1780
theory	0.13712	1.7876	1.7876	1.7876	1.7876	23.1781	23.1781	23.1781	23.1781

$n/2 - 1$ are matched pairs. Thus the average percentages for the matched pairs are made with a smaller population and may deviate (slightly) more from theory than the average for all pairs. Here the deviation from theory is within the tolerance given the sample of size.

6.2 Wave Signatures without Rejection Vectors

Below the percentage for each value of (a, b) on average over all pairs and on average over all matched pairs. Those values were obtained from a sample of 100 000 signatures produced by our SageMath implementation of Wave in which all decoding results are accepted (*i.e.* stage 3 is completely cancelled). The statistics for the matched pair are very close to the expectation

Table 2: Wave without Stage 3

(a, b)	$(0, 0)$	$(1, 0)$	$(2, 0)$	$(0, 1)$	$(0, 2)$	$(1, 1)$	$(2, 1)$	$(1, 2)$	$(2, 2)$
matched	0.14259	1.7847	1.7844	1.7846	1.7856	23.1730	23.1867	23.1839	23.1744
all	0.13712	1.7871	1.7871	1.7880	1.7881	23.1773	23.1784	23.1779	23.1790
theory	0.13712	1.7876	1.7876	1.7876	1.7876	23.1781	23.1781	23.1781	23.1781

for a uniform distribution. However small deviations can be observed. Note that using thoses

deviations to produce an attack won't be easy, several order of magnitude harder than those described below when the stage 2 is incorrectly implemented. Still, with the distributions we chose here (truncated Laplace) it is unsafe to make the economy of stage 3.

6.3 Raw $(U, U+V)$ Decoder Output

Below the percentage for each value of (a, b) on average over all pairs and on average over all matched pairs. Those values were obtained from a sample of 1 200 signatures produced from the Barreto Persichetti Magma script at the end of [1]. As expected, we observe a huge bias,

Table 3: Raw Decoder

(a,b)	$(0,0)$	$(1,0)$	$(2,0)$	$(0,1)$	$(0,2)$	$(1,1)$	$(2,1)$	$(1,2)$	$(2,2)$
matched	1.24320	1.2236	1.2286	1.2324	1.2488	16.0742	30.8580	30.8703	16.0209
all	0.13706	1.7857	1.7835	1.7887	1.7878	23.1986	23.1704	23.1907	23.1575
theory	0.13712	1.7876	1.7876	1.7876	1.7876	23.1781	23.1781	23.1781	23.1781

especially in the last 4 columns. In [1] the observed statistic is $\mathbb{P}(e_i = -e_j) - \mathbb{P}(e_i = e_j)$. In other words, one counts for a given position pair (i, j) the number of occurences of the values $(1, 2)$ or $(2, 1)$ minus the number of occurences of the values $(1, 1)$ or $(2, 2)$. The resulting number should be 0 on average (columns $(2, 1)$ and $(1, 2)$ minus columns $(1, 1)$ and $(2, 2)$) for a random pair, and almost 30% ($\approx 30.85 + 30.87 - 16.07 - 16.02$) of the sample size on average when the pair is matched. This will reveal the matched pairs even with a small sample size. Note that this bias is completely absent in the Wave signatures.

6.4 Tsunami Signatures

Below the percentage for each value of (a, b) on average over all pairs and on average over all matched pairs. Those values were obtained from a sample of 1 200 Tsunami signatures produced from the Barreto Persichetti Magma script at the end of [1]. The bias in the last columns

Table 4: Tsunami

(a,b)	$(0,0)$	$(1,0)$	$(2,0)$	$(0,1)$	$(0,2)$	$(1,1)$	$(2,1)$	$(1,2)$	$(2,2)$
matched	1.85141	0.9251	0.9403	0.9301	0.9239	24.0577	23.1444	23.1529	24.0741
all	0.13716	1.7884	1.7884	1.7849	1.7863	23.1684	23.1701	23.1879	23.1885
theory	0.13712	1.7876	1.7876	1.7876	1.7876	23.1781	23.1781	23.1781	23.1781

appears to be corrected in Tsunami signatures. Interestingly, the bias seems to amplify in the first 5 columns. For instance, the position pairs (i, j) in Tsunami signatures whose values are most frequently equal to zero simultaneously are likely to be matched.

References

[1] Paulo S. L. M. Barreto and Edoardo Persichetti. Cryptanalysis of the Wave signature scheme. Cryptology ePrint Archive, Report 2018/1111, 2018.

[2] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Surf: a new code-based signature scheme. preprint, September 2017. arXiv:1706.08065v3.

[3] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. Cryptology ePrint Archive, Report 2018/996, v4, May 2019. `https://project.inria.fr/wave`