About Wave Implementation and its Leakage Immunity*

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

[1] Sorbonne Université, Paris , France
[2] Inria, Paris, France
{thomas.debris,nicolas.sendrier,jean-pierre.tillich}@inria.fr

October 29, 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 and in C. It allowed us to check that properly generated Wave signatures are uniformly distributed.

Preliminary Statements. We consider here an early version of Wave (v1 of [3]). This version already had all the features guarantying the absence of leakage. It uses 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 the 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.

The description of the sampling stages (§2) is also valid for v2. After that, the gap d was introduced and stage 2 uses $k_U - d$ and $k_V - d$ instead of k_U and k_V. The rest of the paper remains essentially true for all versions of Wave, but the statistics of §5 are only relevant for strict $(U, U+V)$ codes. For generalized $(U, U+V)$ codes coefficients must be applied, but similar arguments could be produced.

A reference implementation is available at the following URL: `http://wave.inria.fr`.

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 $f(x) = y$.

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

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.

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].

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

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.

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]) 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 §5.

4 Implementing Wave

As proof a proof of concept, we implemented Wave as described in §2. This was done in SageMath and in C 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. Moreover, since the distributions of stage 2 are discretized, truncated, rounded to finite precision, and possibly shifted to minimize rejection (see [3]) a deviation of the mean value may occur. This may provoke a small bias and explain the tiny deviation observed in Table 2. Note that this deviation will be corrected by the rejection vectors.

[2] `http://wave.inria.fr`

3. The acceptance rate is exponentially small if distributions are not well chosen. It is the case for instance by choosing 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 SageMath version uses the native floating point arithmetic to compute the rejection vectors. This could result in a small leakage. The rejection vectors of the C implementation were generated with a multi-precision arithmetic and do not have this potential problem. Nevertheless, what we present here, though it was computed from the SageMath implementation, is enough to judge of the feasibility of a leakage-free implementation.

5 Measured Statistics

The tested three categories of signatures. First genuine Wave signatures. Next Wave signatures in which the stage 3 is not applied. The last one was produced from the Magma script of [1]. It implements the "raw decoder" (without the rejection sampling feature). Though it is incorrect to claim an attack from those signatures (and the paper [1] was later withdrawn because of that), the statistics are interesting to illustrate the need for rejection sampling.

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 mentioned 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. 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.

5.1 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 $n/2 - 1$ 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.

(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

Table 1: Wave

5.2 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

(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

Table 2: Wave without Stage 3

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

5.3 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 using an unmodified decoder. As expected, we observe a huge bias, especially in the last 4 columns. In [1] the observed

(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

Table 3: Raw Decoder

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 occurrences of the values $(1,2)$ or $(2,1)$ minus the number of occurrences 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.

References

[1] Paulo S. L. M. Barreto and Edoardo Persichetti. Cryptanalysis of the Wave signature scheme. Cryptology ePrint Archive, Report 2018/1111, 2018. Withdrawn July 2019.

[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.