W-OTS# - Shorter and Faster Winternitz Signatures


```
Author: Herman Schoenfeld <herman@sphere10.com>
Version: 1.0
Date: 2020-07-20
```


Abstract

A very simple modification to the standard W-OTS scheme is presented called W-OTS# that achieves a security enhancement similar to W-OTS+ [1] but without the overhead of hashing a randomization vector in every round of the chaining function. The idea proffered by W-OTS# is to simply thwart Birthday-attacks [2] altogether by signing an HMAC of the message-digest (keyed with cryptographically random salt) rather than the message-digest itself. The signer thwarts a birthday attack by virtue of requiring that the attacker guess the salt bits in addition to the message-digest bits during the collision scanning process. By choosing a salt length matching the message-digest length, the security of W-OTS# reduces to that of the cryptographic hash function. This essentially doubles the security level of W-OTS and facilitates the use of shorter hash functions which provide shorter and faster signatures for same security. For example, W-OTS# 128-bit signatures have commensurate security to standard W-OTS 256-bit signatures yet are roughly half the size and twice as fast. It is proposed that `Blake2b-128` and Winternitz parameter `w=4` (i.e. base-16 digits) be adopted as the default parameter set for the W-OTS# scheme.

1. Birthday Attack

A birthday attack involves an attacker forging a signature for a "malicious" message `M` by re-using a signature for an "agreed" message `m`. In this class of attack, the attacker has pre-knowledge of a message `m` that the victim is willing and intending to sign in the future.

The attacker creates variations of `m` as `{m_1..m_k}` any of which will also be deemed "valid" and signed by the victim. Whilst the victim considers each message `m_i` "identical", their hash digests are unique. This can be achieved by simply varying nonces or whitespace within `m` to create this set.

The attacker simultaneously generates variations of a "malicious" message `M` as the set `{M1..M_l}` and stops until a collision `H(m_i) = H(M_j)` is found (where `H` is the hash function used in the scheme).

Note the probability of finding such collisions is far more likely than a standard brute-force attack by virtue of the Birthday problem [2] [3].

When a collision-pair `(m_i, M_j)` is found, the attacker asks the victim to sign valid `m_i` giving `s = Sign(m_i, key) = SignDigest(H(m_i), key)`. The attacker then proceeds to forge a signature for invalid `M_i` by simply re-using `s`, as follows:

```
S = Sign(M_j, key)
  = SignDigest(H(M_j), key)
  = SignDigest(H(m_i), key)
  = S
```

Unbeknownst to the victim, by signing `m_i`, they have also signed `M_j`.

2. W-OTS & W-OTS+

The Winternitz scheme is a well-documented[4][5] scheme whose description is beyond the scope of this document. However, of relevance is the relationship between the W-OTS "security parameter" `n` (the bit-length of `H`) and it's "security level" which is generally `n/2`. This follows from the fact that if a brute-force attack on `H` requires `2^n` hash rounds then a birthday attack requires `2^(n/2)`[2] hash rounds. By eliminating the birthday attack, and assuming no such other class of attacks exist for `H`, the security level of the scheme is restored back to that of a brute-force attack on `H` which is `n`.

W-OTS+ achieves a similar security enhancement through obfuscation of pre-images in the hashing chains, however they are performed during the chaining function which adds an overhead (significant in some implementations). W-OTS# is similar to W-OTS+ in this regard except it only obfuscates the message-digest once via an HMAC (keyed with the salt) and uses the standard W-OTS chaining function, which is faster than W-OTS+. Despite the concatenation of the salt to the signature, the overall signature size decreases by virtue of selecting a shorter hash function `H`.

3. W-OTS#

The W-OTS# construction is identical to a standard W-OTS construction for Winternitz parameter `w` and cryptographic hash function `H`. The security parameter `n` is inferred from the the bit-length of `H`.

In W-OTS, a message-digest `md` is computed as `md=H(message)`. During signing, digits of base `2^w` are read from `md` and signed in a Winternitz chain. In W-OTS#, the message-digest `md` is replaced with the "sig-mac" `smac` defined as:

3.1 Signature Message Authentication Code (SMAC)

```
smac = SMAC(m, salt)
     = HMAC(H(m), salt)
     = H(Salt || H(Salt || H(m)))
```

The `salt` is concatenated to the signature and used to compute `smac` during verification.

NOTE the checksum digits are calculated and signed identically as per W-OTS but derived from `smac` not `md`.

3.2 Salt

The `Salt` is generated by the signer using cryptographic random number generator. The length of the `Salt` is `n` bits which is the minimum value required to nullify a birthday attack (proven below). The salt is defined as:

```
1: Salt = {0,1}^n   (i.e. n cryptographically random bits)
```

3.1.2 Proof

1. A birthday-collision is expected after `1.25 * SQRT(U)` [2] hashing rounds where `U` is maximum hashing rounds ever required (non-repeating).
2. In W-OTS, `U=2^n` where `n` is the security parameter (bits-length of `H`) and thus (1) becomes `1.25 * 2^(n/2)`.
3. In W-OTS#, adding a `d`-bit salt hardens a birthday-collision to `A = 1.25 * 2^((n+d)/2)` rounds. This follows from the fact that an attacker must scan for collision `(HMAC(H(m_i), Salt), HMAC(H(M_j), Salt))` which involves `d` more bits (whereas in W-OTS they just scan for `(H(m_i), H(M_j))`).
4. A brute-force attack on `H` requires `B = 2^n` hashing rounds [2].
5. We need to choose `d` such `A = B`, since we only need to harden a birthday attack to match that of a brute-force attack. Hardening beyond is redundant since the security level of the scheme is only as strong as the weakest attack vector.
6. Evaluating (5) gives `d = 2 ln(0.8)/ln(0.2) + n = 0.2773 + n` which is approximately `n`
7. Thus choosing `d=n` is sufficient to thwart birthday-attack. QED.

4. References

1. Hülsing, A. "W-OTS+ -Shorter Signatures for Hash-Based Signature Schemes". 2013. Url: https://eprint.iacr.org/2017/965.pdf. Accessed: 2020-07-22. ↩

2. Wikipedia. "Birthday Attack". Url: https://en.wikipedia.org/wiki/Birthday_attack. Accessed: 2020-07-22 ↩ ↩ ↩ ↩ ↩

3. Wikipedia. "Birthday Problem". Url: https://en.wikipedia.org/wiki/Birthday_problem. Accessed: 2020-07-22 ↩

4. Ralph Merkle. "Secrecy, authentication and public key systems / A certified digital signature". Ph.D. dissertation, Dept. of Electrical Engineering, Stanford University, 1979. Url: http://www.merkle.com/papers/Certified1979.pdf ↩

5. Crypto4A. "Hash Chains and the Winternitz One-Time Signature Scheme". URL: https://crypto4a.com/sectorization-defunct/wots/. Accessed on: 2020-07-20 ↩