XMSS trivial encoding

The views expressed herein are exclusively those of Tom Wambsgans, and do not engage any other party.

1 Context

The current implementation does not use the suggested encoding of [1], which leads to a theoretically optimal number of hashes / signature, but is more challenging to arithmetize (and would probably require additional precompiles in the zkVM).

In order to make progress on the rest of the zkVM, we temporarily use what we could call the "trivial encoding". **Note that this hasn't been approved by anyone with relevant expertise**, and should be seen as a toy example. We (non-expert) still believe this encoding is secure, in the Random Oracle Model (and thus may potentially be secure with less optimistic hypotheses).

2 Describtion of the encoding

The encoding is designed for Target-Sum XMSS.

We use the notations:

- v: number of hash chains
- w: length of each hash chain

An encoding is a tuple $(e) \in \{0, \dots, w-1\}^v$ such that $\sum_{i=0}^{v} e_i = t$, where t is the target sum, and $d = v \cdot (w-1) - t$ is the associated distance.

More details about XMSS signatures can be found in any of these 3 articles: [2], [3] and [1].

The trivial encoding works as follows:

1. Hash the message, and the randomness (included in the signature). In our case, we use Poseidon over the prime field koala-bear ($p = 2^{31} - 2^{24} + 1$), the hash output is thus in $(\mathbb{F}_p)^n$.
2. Extract in each one the n field elements 24 bits. More precisely, for $x = \sum_{i=0}^{31} x_i \cdot 2^i \in \mathbb{F}_p$, take $(x_0, \dots, x_{23})$.

3. Use every sequence of 24 bits to form $z = 24/\log_2(w)$ numbers in $\{0, \ldots, w-1\}$.
4. Concatenate everything to obtain v such numbers.
5. If their sum does not equal the target t, reject.

3 Instantiation

We suggest $v = 68$, $w = 4$ and $d = 90$.

For a lifetime of 2^{32}, we get:

- $90 + 1 + 32 + 34 = 157$: hashes / XMSS verification.
- $(68 + 1 + 32) \cdot 31$ bytes $= 3.06$ KiB: size of a signature

Additional details:

- The number of possible encoding is

$$\ell_d = \sum_{i=0}^{\lfloor t/w \rfloor} (-1)^i \binom{v}{i} \binom{t - w \cdot i + v - 1}{v - 1} > 2^{130}$$

  .
- On average, the algorithm above requires ≈ 54 iterations to produce a valid encoding.

4 Security

We assume the ROM (Random Oracle model) idealization for our hash function.

4.1 Classical security

- Each sequence of 24 bits extracted from a field element appears with same probability $p_0 = 126/((2^{24} - 1) \cdot 126 + 127)$, except for $(1, 1, \ldots, 1)$ which appears with probability $p_1 = \frac{127}{126} \cdot p_0$
- The encoding consists in v/z groups of z numbers in $\{0, \ldots, w-1\}$. Among each group of z numbers, all the combinations appears with the same probability p_0, except for $(w-1, \ldots, w-1)$ which appears with probability p_1.
- The probability that the algorithm described in 2 outputs a given encoding (e) is $p_e = p_0^{v/z-c} \cdot p_1^c$ where c is the number of occurrences of $(w-1, \ldots, w-1)$ among the v/z groups of size z in (e).

- There at most $\frac{v}{2z}$ such occurrences (as long as the target sum is "above average"). As a consequence, we have a bound on the probability of each encoding (e):

$$p_e \leq p_0^{v/(2z)} \cdot p_1^{v/(2z)} = (127/126)^{v/(2z)} \cdot p_0^{v/z}$$

$$< (127/126)^{v/(2z)} \cdot (1/2^{24})^{v/z} = (127/126)^{v/(2z)} \cdot \frac{1}{2^{v \cdot \log_2(w)}}$$

- In our case, $p_e \leq 1/2^{135}$

  As a result, in the ROM, an attacker would have to run $\approx 2^{135}$ times the algorithm in 2 (iterating over the set of message / randomness) before finding a collision with an already signed encoding.

  Hence the ≥ 128 bits of classical security.

4.2 Quantum security

In the quantum-accessible random oracle model, we get 64 bits of security, according to Theorem 8 of [4].

The intuitive argument, is that, in the ROM, attacking the encoding can be reinterpreted as a "needle in haystack" problem, but this time with multiple needles. The haystack is the set of message / randomness, and the needles are the subset for which the encoding collides with a previously signed one. Given the fact that the fraction of needles is $< 1/2^{135}$, Theorem 8 of [4] states that at least $\sin(\pi/8) \cdot 2^{135/2} \geq 2^{64}$ steps are required, on average, for any quantum algorithm.

References

[1] J. Drake, D. Khovratovich, M. Kudinov, and B. Wagner, "Technical note: LeanSig for post-quantum ethereum," Cryptology ePrint Archive, Paper 2025/1332, 2025. [Online]. Available: https://eprint.iacr.org/2025/1332

[2] ——, "Hash-based multi-signatures for post-quantum ethereum," Cryptology ePrint Archive, Paper 2025/055, 2025. [Online]. Available: https://eprint.iacr.org/2025/055

[3] D. Khovratovich, M. Kudinov, and B. Wagner, "At the top of the hypercube – better size-time tradeoffs for hash-based signatures," Cryptology ePrint Archive, Paper 2025/889, 2025. [Online]. Available: https://eprint.iacr.org/2025/889

[4] M. Boyer, G. Brassard, P. Høyer, and A. Tapp, "Tight bounds on quantum searching," p. 493–505, Jun. 1998. [Online]. Available: https://arxiv.org/pdf/quant-ph/9605034