Forging tropical signatures

Lorenz Panny

Technische Universität München, Germany
lorenz@yx7.cc

Abstract. A recent preprint [3] suggests the use of polynomials over a tropical algebra to construct a digital signature scheme "based on" the problem of factoring such polynomials, which is known to be NP-hard. This short note presents two very efficient forgery attacks on the scheme, bypassing the need to factorize tropical polynomials and thus demonstrating that security in fact rests on a different, empirically easier problem.

Keywords: Cryptanalysis, tropical algebra, digital signatures, factorization.

1 Introduction

Tropical algebra is concerned with the *min-plus semiring*, or *tropical semiring*: an algebraic structure over the extended reals defined by the two binary operations "min" and "+". Multiple candidate cryptographic constructions using the tropical semiring in various ways have been suggested in a series of papers starting about ten years ago, and, to a large extent, broken. Readers may wish to consult the introduction of [2] for an overview of past proposals and corresponding attacks.

The most recent proposal [3] in the tropical-cryptography family is a signature scheme building on multiplication of polynomials over the tropical semiring. In this note, we give two fairly simple, yet very effective forgery attacks against the scheme and show that the attacker is in fact not required to solve the claimed hard problem underneath the construction. We provide implementations of both attacks, confirming that they work in practice.

These results also highlight the need for clear problem statements and unambiguous claims concerning the relationships between them: The signature scheme of [3] may be "based on" the factorization problem for tropical polynomials in the sense that it is *inspired by* that problem. However, there is no known result indicating that breaking the signature scheme is as hard as factoring tropical polynomials (e.g., a reduction from that factoring problem to breaking the EUF-CMA security of the scheme), and indeed, our attacks strongly suggest the opposite.

As a side remark, we also note that the emphasis of [3] on the NP-hardness of factoring tropical polynomials misses the point: Cryptography requires computational problems that are hard *for almost all instances*, while the notion of NP-hardness only says something about the hardest, possibly very few instances. See the classic paper [5] for an informal overview of average-case complexity.

[*] Date of this document: 2024-01-13.

Parallel and subsequent work. Soon after this note first appeared online, a larger set of attacks against [3] and some countermeasures (insufficient to prevent all known attacks) were presented in [1]. Yet another attack which was not covered by [1] was presented in [6]; it bypasses the countermeasures of [1] too.

1.1 Tropical polynomials

The *tropical semiring* is defined as the set $\mathbb{R} \cup \{+\infty\}$ with two binary operations $a \oplus b = \min\{a, b\}$ and $a \otimes b = a + b$. With this, *tropical polynomials* are simply symbolic expressions of the form

$$f(x) = c_0 \oplus (c_1 \otimes x) \oplus (c_2 \otimes x^{\otimes 2}) \oplus \cdots \oplus (c_n \otimes x^{\otimes n})$$

where x is a variable and the coefficients $c_0, \ldots, c_n$ lie in the tropical semiring; they too form a semiring under the operations $\oplus$ and $\otimes$ extended to polynomials in the usual way. Note that "missing" coefficients are interpreted as $+\infty$.

Example 1. In more usual notation, $f(x) = \min\{c_0, c_1 + x, c_2 + 2x, \ldots, c_n + nx\}$.

Example 2. The tropical polynomial product $f(x) \otimes g(x)$ has coefficients $c_0 + c'_0$, $\min\{c_0 + c'_1, c_1 + c'_0\}$, $\min\{c_0 + c'_2, c_1 + c'_1, c_2 + c'_0\}$, etc., where c_i and c'_j are the coefficients of $f(x)$ and $g(x)$ respectively. Generally, the k^{th} coefficient equals

$$\min\left\{c_i + c'_{k-i} : i \in \{0, \ldots, k\}\right\}.$$

Remark 3. Beware that there are at least two notions of "tropical polynomial" in the literature: One is the algebraic viewpoint employed in [3] and here. The other identifies polynomials with the *functions* they define on $\mathbb{R}$. In the latter formalism, all tropical polynomials split uniquely into linear tropical polynomials and the factors can be found in polynomial time; see for instance [4].

1.2 Signatures from tropical polynomials

The signature scheme proposed by [3] features two integer parameters d and r; the values suggested in [3, § 4] are $d = 150$ and $r = 127$. Let $T_{d,r}$ denote the set of tropical polynomials of degree d and with all coefficients in $\{0, \ldots, r\} \subseteq \mathbb{Z}$, and let H be a collision-resistant hash function from bit strings to $T_{d,r}$.

The public key is a tropical product $M = X \otimes Y$, where (X, Y) are tropical polynomials sampled uniformly from $T_{d,r}$. A valid signature for a message m is a triple (S_1, S_2, N) of tropical polynomials with $S_1, S_2 \in T_{3d,3r}$ and $N \in T_{2d,2r}$, such that $S_1 \otimes S_2 = P \otimes P \otimes M \otimes N$ where $P = H(m)$; in addition, neither S_1 nor S_2 may be a constant (tropical) multiple of $P \otimes M$ or $P \otimes N$.

For the sake of completeness, here is how [3] computes such a signature using the private key $(X, Y) \in T_{d,r} \times T_{d,r}$: Sample $U, V \leftarrow T_{d,r}$ and let $N := U \otimes V$. The signature is $(P \otimes X \otimes U,\ P \otimes Y \otimes V,\ N)$.

Remark 4. By definition (cf. Example 2), multiplying a tropical polynomial by a constant amounts to shifting each coefficient by that constant. Hence, constant multiples can easily be detected using ordinary subtraction on the coefficients.

2 The attacks

The problem of recovering a secret key for a given public key really is factoring (with some extra constraints on the degree and coefficient sizes). However, the problem of forging a signature gives significantly more power to the attacker: Their task is to factor *when another factorization is already known!*

In more detail, recall that a signature consists of a triple (S_1, S_2, N) of tropical polynomials such that $S_1 \otimes S_2 = P \otimes P \otimes M \otimes N$, where M is the public key and P is a hash value of the message encoded as a tropical polynomial in a suitable way. Setting $S_1 = P \otimes M$ and $S_2 = P \otimes N$ solves the verification equation but clearly does not require knowledge of the secret; hence, this pair and its constant (tropical) multiples are excluded as "trivial forgeries" in [3]. However, in tropical algebra, polynomial multiplication is highly non-cancellable, and a tropical polynomial typically has a very large number of distinct product decompositions. All attacks explained in the sequel are based on different methods of discovering new product decompositions starting from a given one.

Note that we focus on the most basic version of each attack for simplicity, and that the forged signatures produced by those methods look rather different from honestly generated signatures. We can (and will) however speculate that any particular method of detecting forgeries can be fooled by carefully combining and adjusting the basic forgery techniques. Indeed, since chaining these basic strategies may prove useful, it is helpful to view them not just as one-shot forgery attacks, but as *transformations* on pairs (S_1, S_2) of tropical polynomials which produce another pair (S_1', S_2') such that $S_1' \otimes S_2' = S_1 \otimes S_2$, while preserving superficial properties like the degrees and bounds on the coefficients.

2.1 Warmup: "Trivial forgeries"

The most trivial forgery would be $(P \otimes M,\ P \otimes N,\ N)$ for any N with the correct degree and coefficient bounds. A mild generalization is to (tropically) multiply the three components by constants in such a way that the verification equation and coefficient bounds remain satisfied; this yields signatures of the form $(c_1 \otimes P \otimes M,\ c_2 \otimes P \otimes N,\ c_1 \otimes c_2 \otimes N)$. Forgeries of this type are detected and rejected by the verification algorithm specified in [3]. However, they form a big part of the following stronger attacks in the sense that they provide the initial product decomposition $(S_1, S_2) = (P \otimes M,\ P \otimes N)$ of $P \otimes P \otimes M \otimes N$.

2.2 Attack #1: Morphing products

The starting point for the first attack is to recall Example 2 and notice that there will almost certainly be coefficients c_i and c_j' in the inputs (S_1, S_2) which effectively do not contribute to the result at all: Only the smallest sum $c_i + c_{k-i}'$ in each coefficient of the product matters; the remaining information disappears under the minimum. This phenomenon is particularly pronounced for the middle coefficients, where each minimum runs over a large number of individual sums $c_i + c_j'$.

Hence, the idea is to search for small modifications to the coefficients of both $S_1 = P \otimes M$ and $S_2 = P \otimes N$ which do not spoil the final product. The simplest conceivable approach is to first find a single coefficient c_i in S_1 such that setting $c_i \leftarrow c_i \pm 1$ preserves the product with S_2, and then find a single coefficient c'_j in S_2 such that setting $c'_j \leftarrow c'_j \pm 1$ preserves the product with that modified S_1. Experimentally, for the parameter choices suggested in [3], there appears to be plenty of freedom in S_1 and S_2 to produce a forgery that passes verification.

We stress again that there is no reason to stop at modifying a single coefficient in each value by ± 1 only; hence, the obvious countermeasure of banning values of S_1 and S_2 which are "close to" $P \otimes M$ or $P \otimes N$ is insufficient.

2.3 Attack #2: Swapping divisors

The second attack is based on the simple observation that divisors of S_1 can be moved over to S_2, and vice versa, without changing the value of the product $S_1 \otimes S_2$. Notably, a full factorization of the respective polynomials is not needed, and finding small-degree divisors appears to be significantly easier — if they exist. Empirically, generic constraint solvers are capable of computing (or ruling out) divisors of reasonably small degree very quickly; see Section 2.4.

Forging signatures thus works as follows: Search for a message such that $P \otimes M$ can be decomposed as $D_1 \otimes R_1$ with $\deg(D_1) \ll \deg(R_1)$. Once found, search for a nonce N as in the legitimate signing algorithm that can be decomposed as $D_2 \otimes R_2$ with $\deg(D_2) = \deg(D_1)$. The signature is $(D_1 \otimes R_2,\, D_2 \otimes R_1,\, N)$. In practice, executing this forgery becomes a little easier by directly assembling N with a suitable pre-existing known factor, rather than searching randomly.

2.4 Attack implementation

We provide Python code for both attack varieties, using the proof-of-concept implementation of [3] to check that forgeries were successful. While the script for attack #1 is entirely elementary, the script for attack #2 makes use of the Python interface to the *z3* solver [7] to find divisors of tropical polynomials up to degree 5. (This cutoff was chosen pretty arbitrarily.)

Both attacks require no more than a few seconds of computation on average.

Code: https://yx7.cc/files/tropical-attack.tar.gz

References

[1] Daniel R. L. Brown and Chris Monico. *More forging (and patching) of tropical signatures.* IACR Cryptology ePrint Archive 2023/1837. 2023. URL: https://ia.cr/2023/1837.

[2] Ivan Buchinskiy, Matvei Kotov and Alexander Treier. *Analysis of four protocols based on tropical circulant matrices.* IACR Cryptology ePrint Archive 2023/1707. 2023. URL: https://ia.cr/2023/1707.

[3] Jiale Chen, Dima Grigoriev and Vladimir Shpilrain. *Tropical cryptography III: digital signatures*. IACR Cryptology ePrint Archive 2023/1475. 2023. URL: https://ia.cr/2023/1475.

[4] Nathan Grigg and Nathan Manwaring. *An Elementary Proof of the Fundamental Theorem of Tropical Algebra*. 2007. arXiv: 0707.2591.

[5] Russell Impagliazzo. "A personal view of average-case complexity". In: *Proceedings of Structure in Complexity Theory*. 1995, pp. 134–147.

[6] Minsun Kim. *"Tropical Santa" (2023 Christmas CTF)*. Blog post. URL: https://soon.haari.me/2023-christmas-ctf/#tropical-santa.

[7] Leonardo de Moura and Nikolaj Bjørner. "Z3: An Efficient SMT Solver". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340. URL: https://github.com/Z3Prover/z3.