TROPICAL CRYPTOGRAPHY IV: DIGITAL SIGNATURES AND SECRET SHARING WITH ARBITRARY ACCESS STRUCTURE

DIMA GRIGORIEV, CHRIS MONICO, AND VLADIMIR SHPILRAIN

ABSTRACT. We use tropical algebras as platforms for a very efficient digital signature protocol. Security relies on computational hardness of factoring a given tropical matrix in a product of two matrices of given dimensions; this problem is known to be NP-complete. We also offer a secret sharing scheme with an arbitrary access structure where security of the shared secret is based on computational hardness of the same problem.

1. INTRODUCTION

In [7], [8], we employed *tropical algebras* as platforms for cryptographic schemes by mimicking some well-known classical schemes, as well as newer schemes like [9], in the "tropical" setting. What it means is that we replaced the usual operations of addition and multiplication by the operations $\min(x, y)$ and $x + y$, respectively.

An obvious advantage of using tropical algebras as platforms is unparalleled efficiency because in tropical schemes, one does not have to perform any multiplications of numbers since tropical multiplication is the usual addition, see Section 2. The focus therefore is entirely on security. Several "tropical" protocols were attacked (see e.g. [1], [2], [11]) by showing that one or another problem known to be hard in the worst case, turns out to be easy for a non-negligible set of inputs.

In this paper, we use a tropical algebra of matrices to design a digital signature scheme and a secret sharing scheme with an arbitrary collection of secret-recovering coalitions. Security of the private keys in these schemes is based on computational hardness of factoring a given tropical matrix in a product of two matrices of given dimensions. This problem is known to be NP-complete, see [13].

2. PRELIMINARIES

We start by giving some necessary information on tropical algebras here; for more details, we refer the reader to the monograph [3].

Consider a tropical semiring S, also known as the min-plus algebra due to the following definition. This semiring is defined as a linearly ordered set (e.g., a subset of reals) that contains 0 and is closed under addition, with two operations as follows:

$x \oplus y = \min(x, y)$

$x \otimes y = x + y.$

It is straightforward to see that these operations satisfy the following properties:

associativity:
$x \oplus (y \oplus z) = (x \oplus y) \oplus z$
$x \otimes (y \otimes z) = (x \otimes y) \otimes z.$

commutativity:
$x \oplus y = y \oplus x$
$x \otimes y = y \otimes x.$

distributivity:
$(x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z).$

There are some "counterintuitive" properties as well:
$x \oplus x = x$

$x \otimes 0 = x$

$x \oplus 0$ could be either 0 or x.

There is also a special "ϵ-element" $\epsilon = \infty$ such that, for any $x \in S$,

$\epsilon \oplus x = x$

$\epsilon \otimes x = \epsilon.$

2.1. **Tropical matrices.** A tropical algebra can be used for matrix operations as well. To perform the $A \oplus B$ operation, the elements m_{ij} of the resulting matrix M are set to be equal to $a_{ij} \oplus b_{ij}$. The $\otimes$ operation is similar to the usual matrix multiplication, however, every "$+$" calculation has to be substituted by a $\oplus$ operation, and every "$\cdot$" calculation by a $\otimes$ operation.

Example 1. $\begin{pmatrix} 1 & 2 \\ 5 & -1 \end{pmatrix} \oplus \begin{pmatrix} 0 & 3 \\ 2 & 8 \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 2 & -1 \end{pmatrix}.$

Example 2. $\begin{pmatrix} 1 & 2 \\ 5 & -1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 3 \\ 2 & 8 \end{pmatrix} = \begin{pmatrix} 1 & 4 \\ 1 & 7 \end{pmatrix}.$

The role of the identity matrix I is played by the matrix that has "0"s on the diagonal and ∞ elsewhere. Similarly, a scalar matrix would be a matrix with an element $\lambda \in S$ on the diagonal and ∞ elsewhere. Such a matrix commutes with any other square matrix (of the same size). Multiplying a square matrix by a scalar amounts to multiplying it by the corresponding scalar matrix.

Example 3. $2 \otimes \begin{pmatrix} 1 & 2 \\ 5 & -1 \end{pmatrix} = \begin{pmatrix} 2 & \infty \\ \infty & 2 \end{pmatrix} \otimes \begin{pmatrix} 1 & 2 \\ 5 & -1 \end{pmatrix} = \begin{pmatrix} 3 & 4 \\ 7 & 1 \end{pmatrix}.$

Then, tropical *diagonal matrices* have something (but not ∞) on the diagonal and ∞ elsewhere.

We also note that, in contrast with the "classical" situation, it is rather rare that a "tropical" matrix is invertible. More specifically (see [3, p.5]), the only invertible tropical matrices are those that are obtained from a diagonal matrix by permuting rows and/or columns.

Example 4. $\begin{pmatrix} a & \infty \\ \infty & b \end{pmatrix}^{-1} = \begin{pmatrix} -a & \infty \\ \infty & -b \end{pmatrix}.$

Example 5. Conjugation:

$$\begin{pmatrix} a & \infty \\ \infty & b \end{pmatrix}^{-1} \otimes \begin{pmatrix} x & y \\ z & t \end{pmatrix} \otimes \begin{pmatrix} a & \infty \\ \infty & b \end{pmatrix} = \begin{pmatrix} x & y+(b-a) \\ z+(a-b) & t \end{pmatrix}.$$

3. Digital signature scheme description

Here we offer a digital signature scheme that uses the algebra of tropical matrices (over integers) as the platform.

We point out that in [4], we offered a digital signature scheme where security of private keys relied on the NP-hardness of the *tropical polynomial* factorization problem [10]. The original version of that scheme was attacked by Brown and Monico [1] who noticed (experimentally) that, even if the selected private polynomials $P_1(x), P_2(x)$ were of a high degree, there is sometimes a factor of $P(x) = P_1(x) \otimes P_2(x)$ that has a small degree, and then such a small degree factor can be found just by brute force search. One therefore has to be careful with key generation to avoid small degree factors in a product $P_1(x) \otimes P_2(x)$. (We note that factorization of a tropical one-variable polynomial is not necessarily unique [10].)

With factoring tropical matrices, the situation is different: there is a theoretical explanation (see [13]) why there cannot be a "small" tropical matrix factor if the originally selected private factors were not of a small combinatorial rank (a.k.a. Barvinok rank). Therefore, an attack in the spirit of [1] will not work with matrices.

Now the signature scheme is as follows.

Private:
– two matrices X, Y whose dimensions are $m \times k$ and $k \times n$, respectively, where m, k and n are parameters of the scheme.

Public:
– matrix $T = X \otimes Y$.
– a hash function H (e.g., SHA-512) and a (deterministic) procedure for converting values of H to a matrix M of dimensions $n \times m$ (see Section 4.1).

Signing a message s:

S1. Add a time stamp t to s and apply a hash function H to s combined with t. Denote the result by $H(s,t)$. Convert $H(s,t)$ to a matrix M of dimensions $n \times m$ using a deterministic public procedure.

S2. Select private session keys: two matrices U, V whose dimensions are $n \times k$ and $k \times m$, respectively. Denote $P = X \otimes V$, $R = U \otimes Y$, $S = U \otimes V$. Check that $(M \otimes X) \oplus U \neq U$ and $(Y \otimes M) \oplus V) \neq V$; otherwise select different U, V.

S3. The signature of a message s (together with a time stamp t) is the following tuple of matrices:
$(M,\ (M \otimes X) \oplus U,\ (Y \otimes M) \oplus V),\ P,\ R,\ S)$.

Verification:

V1. The verifier computes the hash $H(s,t)$ and converts $H(s,t)$ to a matrix M of dimensions $n \times m$ using a deterministic public procedure. This is done to verify that M is the correct hash of the message.

V2. Denote $Z = (M \otimes X \otimes V) \oplus (U \otimes Y \otimes M)$. Note that the matrix Z can be computed from public information, specifically from the matrices M, P, and R.

V3. The verifier computes $W = [(M \otimes X) \oplus U] \otimes [(Y \otimes M) \oplus V]$. If $W = S = U \otimes V$, the signature is not accepted. The signature is accepted if and only if $W = (M \otimes T \otimes M) \oplus Z \oplus S$.

Correctness follows from $W = ((M \otimes X) \otimes (Y \otimes M)) \oplus (M \otimes X \otimes V) \oplus (U \otimes Y \otimes M) \oplus (U \otimes V) = (M \otimes S \otimes M) \oplus Z \oplus S$.

4. Key generation and suggested parameters

The suggested parameter values are:
$k = 7$
$m = n = 8$
Entries of participating matrices are selected as follows:
– for matrices X, Y: uniformly at random from the range [0, 255].
– for matrices U, V: uniformly at random from the range [0, 255].
– for matrix M: entries are in the range [0, 255]; they are determined from the hash $H(s, t)$, see Section 4.1.

4.1. **Converting $H(s, t)$ to a tropical matrix over Z.**

We suggest using a hash functions from the SHA-3 family, specifically SHA3-512. We assume the security properties of SHA3-512, including collision resistance and preimage resistance. We also assume that there is a standard way to convert $H(s, t)$ to a bit string of length 512. Then a bit string can be converted to a tropical matrix over **Z** using the following *ad hoc* (deterministic) procedure.

Let $B = H(s, t)$ be a bit string of length 512. We will convert B to a matrix M as follows. For convenience, we are going to assume here that the dimensions of M are 8×8. Then M has 64 entries. We can therefore split the bit string B of length 512 into 64 blocks of length 8, and these blocks, interpreted as the binary form of integers, will be the entries of M (populating M in whatever deterministic order).

5. What is the hard problem here?

The (computationally) hard problem that we employ in our construction is factoring a given tropical $m \times n$ matrix as a product of an $m \times k$ matrix and a $k \times n$ matrix. The computational hardness of this problem is confirmed by the following result:

Theorem 1. [13] The following problem is NP-complete for any $k \geq 7$: given an $m \times n$ tropical matrix M, find out whether or not M is a product of an $m \times k$ matrix and a $k \times n$ matrix.

The problem relevant to our situation is actually a search version of this problem: given that M is a product of an $m \times k$ matrix X and a $k \times n$ matrix Y, find at least one pair (X, Y) of matrices like that. However, it is well known that search in an NP-hard problem is hard, too.

6. Possible attacks

We see several possible avenues that an attacker might consider in order to forge a signature in this scheme.

(1) Factor the public key T into a product of an $m \times k$ matrix X' and a $k \times n$ matrix Y. But we believe a general instance of this problem to be as difficult as the most difficult instances, and therefore intractable with appropriate choices of parameters.
(2) After observing some number of valid signatures, attempt to recover X or Y. If an attacker knows some matrices M_j and $(M_j \otimes X) \oplus U_j$, perhaps there is some non-obvious way to recover X (or similarly for Y). At present, we do not see any viable attack along these lines. But if either X or Y could be recovered, it would immediately lead to a factorization of T since the relevant 'division problem' is easy.
(3) A valid signature consists of a public key T and (M, A, B, P, R, S) satisfying $W \neq S$ and $W = A \otimes B$, where $W = (M \otimes T \otimes M) \oplus (M \otimes P) \oplus (R \otimes M) \oplus S$, and the dimensions of the matrices are as follows:

$$M : n \times m, \quad A : n \times k, \quad B : k \times m, \quad P : m \times m, \quad R : n \times n, \quad S : n \times m.$$

Perhaps an attacker could avoid the factorization problem by choosing P, R and S in such a way as to make it easy to factor W? In fact, the reason for the requirement that $W \neq S$ is this could otherwise be often accomplished by choosing U and V with very small entries, taking $S = U \otimes V$, and then taking P and R to have large entries so that $W = S = U \otimes V$; in that case, the forger would trivially take $A = U$ and $B = V$. Since the protocol explicitly forbids that, perhaps an attacker could attempt to choose P, R and S to yield $W = M \otimes T \otimes M$ or $W = M \otimes P$ or $W = R \otimes M$; however, none of the matrices M, P, or R have the proper dimensions to act as an attacker's proxy for A or B, so this does not seem to help an attacker.

7. Performance and signature size

Suppose that all private matrices X, Y, U, V and the hash matrix M are taken to have entries in $[0, N] \cap \mathbf{Z}$. The dimensions, range of entries, and representation size for the public key and each signature component are then as follows:

Matrix	dimensions	range of entries	rep. size, in bits
T	$m \times n$	$[0, 2N]$	$mn \log_2 2N$
M	$n \times m$	$[0, N]$	$nm \log_2 N$
$(M \otimes X) \oplus U$	$n \times k$	$[0, 2N]$	$nk \log_2 2N$
$(Y \otimes M) \oplus V$	$k \times m$	$[0, 2N]$	$nk \log_2 2N$
P	$m \times m$	$[0, 2N]$	$m^2 \log_2 2N$
R	$n \times n$	$[0, 2N]$	$n^2 \log_2 2N$
S	$n \times m$	$[0, 2N]$	$nm \log_2 2N$

For the parameter values we suggest, $n = m = 8$, $k = 7$, and $N = 255$, this yields a public key T consisting of 576 bits, and signatures consisting of 3248 bits.

The most computationally expensive parts of this protocol are the hashing of the message and the generation of random entries for X, Y, U, and V which require $2(m + n)k \log_2 N$ random bits, or 1792 random bits with the suggested parameter values. Beyond that one tropical product is needed to compute T, requiring no more than mkn additions and mkn

comparisons, or 448 of each with the suggested parameter values. For computing the signature, general choices of U and V satisfy $W \neq U \otimes V$, so we may assume that each tropical product and sum is computed only once. We find that once the message has been hashed and random U, V chosen, the protocol requires no more than $k(6nm + 2m^2 + 2n^2 + n + m)$ addition/comparison operations. With the suggested parameter sizes, this is 4592 16-bit integer operations in total. Even in Python, this is quite fast; our sample implementation in Python running on a MacBookPro i7@3.10GHz takes an average of 0.00054 seconds to generate a public key, 0.00161 second to sign a 1KB message, and 0.00140 seconds to verify such a signature. See [5] for a GitHub repository.

8. Authentication

Authentication is one of the basic building blocks of security. There are various methods to authenticate a user (see e.g. the monograph [14]), but here we focus on authentication through a proof of knowledge (a.k.a. zero-knowledge proof). This method allows for employing various interesting computationally hard problems from graph theory, ring theory, group theory, and other areas of mathematics (see e.g. [6]).

In this section, we employ the tropical matrix factorization problem to offer a simple interactive challenge-response protocol for authentication.

The set-up is as in Section 3. Private keys are two matrices X, Y whose dimensions are $m \times k$ and $k \times n$, respectively. The public key is their product XY. Then the authentication protocol goes as follows. (As usual, it has to be repeated several times to avoid blind guessing.)

A1. Alice publishes a matrix $S = U_1XYU_2$ for some matrices U_1, U_2 of appropriate dimensions.

A2. Bob challenges Alice with either 0 or 1 bit, at random.

A3. **(a)** If the bit is 0, then Alice responds with the matrices U_1, U_2, and Bob verifies that $S = U_1XYU_2$.

(b) If the bit is 1, then Alice responds with the matrices U_1XP and $P^{-1}YU_2$ for a random invertible $k \times k$ matrix P, and Bob verifies that the product of these matrices equals $S = U_1XYU_2$.

We note that the problem of factoring ABC given matrices ABC and B should be hard because when B is the identity matrix, this becomes the usual factorization problem for the matrix AC.

We also remind the reader that the only invertible tropical matrices are those that are obtained from a diagonal matrix by permuting rows and/or columns, see the end of Section 2.1.

We leave it at that; details, including key generation, suggested parameters, and possible attacks, are similar to what we have for digital signatures above.

9. Secret sharing

Secret sharing is one of the classical problems in cryptography. The dealer wants to distribute shares of a secret (e.g. of a password) among n parties so that if all of them get together, they can recover the secret, but no group of less than n parties can do that. Somewhat more generally, a (t, n)-threshold scheme is a method of distributing a secret among n participants in such a way that any of the t participants can recover the secret, but any group of $t-1$ participants cannot.

This problem was solved by Shamir [12], and his elegant and perfectly secure solution is now considered canonical.

A harder problem is where access control (through recovering a secret) is given to specific groups (coalitions) of participants, regardless of the number of parties in a group. In this section, we offer simple and efficient solutions to this problem, first based on tropical matrices (in line with the first part of this paper), and then we offer a general method that can be based on an arbitrary semigroup. For enhanced efficiency we suggest using the additive semigroup of the field $\mathbb{F}_{2^m}$, where the operation is simply bitwise XOR.

In either case, the main issue is how to reduce the total number of keys distributed to participants. In most real-life applications this is not a problem since the number of coalitions that are allowed access is typically not very large, but in theory, the number of coalitions can be exponential in the number of participants.

9.1. **Secret sharing based on tropical matrices.** First we give a simple example to illustrate our approach to this problem.

Example 6. Suppose there are 4 participants P_i, and only the following coalitions should have access to a secret: $\{P_1, P_2\}, \{P_2, P_3\}, \{P_1, P_4\}, \{P_3, P_4\}$. Thus, neither $\{P_1, P_3\}$ nor $\{P_2, P_4\}$ should be able to recover the secret.

To arrange that, the dealer would create several products of private matrices A, B, C, D, E, F and distribute them to the participants as follows. Here we will write just XY instead of $X \otimes Y$ to make it easier on the eyes. Thus, P_1 gets ABC and DE, P_2 gets AB and DEF, P_3 gets A and $CDEF$, and P_4 gets BC and DEF. The whole secret is $ABCDEF$.

It is straightforward to check that the designated coalitions can indeed recover the product $ABCDEF$. The remaining subsets of participants that do not contain any of the designated coalitions are $\{P_2, P_4\}$ and $\{P_1, P_3\}$. In the coalition $\{P_2, P_4\}$, the parties are jointly in possession of the matrices AB, BC, and DEF. To recover $ABCDEF$, the parties would have to either factor AB to get A or factor BC to get C.

In the coalition $\{P_1, P_3\}$, the parties are jointly in possession of the matrices ABC, DE, A, and $CDEF$. To recover $ABCDEF$, the parties would have to either factor ABC to get AB or factor $CDEF$ to get F.

More formally, suppose the coalition $\{P_1, P_3\}$ has somehow computed the matrix $ABCDEF$. Then, since they are also in possession of the matrix ABC, they can recover the matrix DEF. Then, from DEF and DE, they can recover the matrix F, i.e., they can factor the matrix $CDEF$, thus solving a provably hard problem. Actually, they have factored the matrix $CDEF$ knowing the matrix DE. Still, the problem of factoring XYZ given matrices XYZ and Y should be hard because when Y is the identity matrix, this becomes the usual factorization problem for the matrix XZ.

For a general situation, let the secret S be the product $S = A_1 \cdots A_m$, where the dimensions of the matrices A_i make this product possible.

We are going to distribute various matrix products of the form $A_i A_{i+1} \cdots A_k$ among participants $P_1, \ldots, P_n$ so that only specific subsets of participants would be able to recover the secret S. (Of course, if a coalition C of participants can recover S, then so does any superset $C' \supseteq C$ of participants.)

Now let $\{P_{i_1}, \dots, P_{i_k}\}$ be a coalition of k participants. The way we distribute matrix products to these participants is easier to describe as follows. Write down all the letters $A_1, \dots, A_m$ in a row. Then insert dividers between some of the letters, so that the total number of dividers is $(k-1)$ and there is at least one letter A_j between any two dividers. Then, going left to right, we assign to participants products of matrices corresponding to sequences of letters between dividers. Thus, P_{i_1} gets the products of matrices corresponding to the sequence of letters left of the leftmost divider; P_{i_2} gets the products of matrices corresponding to the sequence of letters between the first and second divider from the left, etc.

There is one more rule. Suppose that dividers have already been distributed for coalitions $C_1, \dots, C_j$. Then, when distributing dividers for a coalition C_{j+1}, we are allowed to put a divider between letters A_i and A_{i+1} only if there was no divider between these letters when we did distribution for $C_1, \dots, C_j$.

This rule imposes a restriction (a lower bound) on the number of matrices A_i as a function of the sum of the cardinalities of all (minimal) coalitions we want to give access to the secret S. Specifically, the number of matrices A_i has to be at least $\sum_{i=1}^{k}(|C_i|-1)$, where k is the number of (minimal) coalitions we want to give access to. Thus, for our method to be efficient, the number of coalitions has to be not too large (as a function of the total number n of participants), which is often the case in real-life applications. If, on the other hand, the set of (minimal) coalitions consists, say, of all possible coalitions of t participants with t on the order of $\frac{n}{2}$, then the number of such coalitions as well as the number of matrices A_i will be exponential in n.

Now we have

Proposition 1. With the shares distribution as above:

(1) coalitions $C_1, \dots, C_k$ and their supersets can recover the secret S;

(2) unless some of the participants can factor their assigned matrix, it is infeasible for any coalition C different from any C_i (or a superset thereof) to recover the secret S.

Proof. Part (1) is trivial; it follows immediately from the way the secret shares are distributed.

For part (2), suppose $C = \{P_{i_1}, \dots, P_{i_k}\}$ is *not* among coalitions $C_1, \dots, C_s$ that were given access to S, and no subset of C is. If, among all the matrix products collectively distributed to the participants in the coalition C, at least one factor A_i is missing, then these participants cannot possibly recover $S = A_1 \cdots A_m$.

Now suppose no A_i factor is missing among the matrix products collectively distributed to the participants in C. Then there must be two participants in the coalition C, say P_{i_1} and P_{i_2}, such that there is an overlap in the matrix products assigned to them. For example, P_{i_1} gets A_1A_2 and P_{i_2} gets A_2A_3. Now there are two cases to consider.

(a) No matrix product distributed to the participants in C starts with A_3. Then in any product of matrices distributed to the participants in C, either one (or both) of the A_1, A_3 will be missing or A_2 will occur twice. Then participants in C cannot recover the secret S unless A_2 can be recovered either from A_1A_2 or from A_2A_3.

(b) One of the participants, say P_{i_3}, is assigned a matrix product that starts with A_3, e.g. $A_3A_4A_5$. Then in the coalition $C' = \{P_{i_1}, P_{i_3}, \dots, P_{i_k}\}$ no A_i factor is missing among the matrix products collectively distributed to the participants in C', but this coalition has one less participant than C does. Also, by our assumption, C' is not among the coalitions that were given access to the secret S. An obvious inductive argument completes the proof. □

9.2. **Secret sharing based on an arbitrary semigroup.** Let G be a semigroup, with the operation written additively. Let S be the secret to be distributed to coalitions $C_1, \ldots, C_k$ that are given access to S.

The dealer then distributes, to participants in each coalition C_i, random keys that sum up to S.

If G is large enough compared to the number k of coalitions, then the probability for an illegitimate coalition to recover the secret S is small. However, the advantage of the "tropical" scheme in Section 9.1 is that obtaining the secret by an illegitimate coalition is equivalent to solving a provably hard problem (of factoring a tropical matrix).

References

[1] D. R. L. Brown, C. Monico, *More forging (and patching) of tropical signatures*, `https://eprint.iacr.org/2023/1837`

[2] I. Buchinskiy, M. Kotov, A. Treier, *An attack on a key exchange protocol based on max-times and min-times algebras*, Indian J. Pure Appl. Math. 56 (2025), 180–190.

[3] P. Butkovic, *Max-linear systems: theory and algorithms*, Springer-Verlag London, 2010.

[4] J. Chen, D. Grigoriev and V. Shpilrain, *Tropical cryptography III: digital signatures*, J. Math. Cryptology **18** (2024), paper No. 20240005.

[5] GitHub repository: `https://github.com/AssociateDeadWood/gs_trop_sigs`

[6] D. Grigoriev and V. Shpilrain, *Authentication schemes from actions on graphs, groups, or rings*, Ann. Pure Appl. Logic. **162** (2010), 194–200.

[7] D. Grigoriev, V. Shpilrain, *Tropical cryptography*, Comm. Algebra. **42** (2014), 2624–2632.

[8] D. Grigoriev and V. Shpilrain, *Tropical cryptography II: extensions by homomorphisms*, Comm. Algebra. **47** (2019), 4224–4229.

[9] M. Habeeb, D. Kahrobaei, V. Shpilrain, *A secret sharing scheme based on group presentations and the word problem*, Contemp. Math., Amer. Math. Soc. **582** (2012), 143–150.

[10] K. H. Kim, F. W. Roush, *Factorization of polynomials in one variable over the tropical semiring*, `https://arxiv.org/pdf/math/0501167.pdf`

[11] A. Ponmaheshkumar, M. Kotov, R. Perumal, *Cryptanalysis of a key exchange protocol based on a digital semiring*, Comm. Algebra **53** (2025), 5226–5236.

[12] A. Shamir, *How to share a secret*, Comm. ACM **22** (1979), 612-–613.

[13] Ya. Shitov, *The complexity of tropical matrix factorization*, Adv. Math. **254** (2014), 138–156.

[14] R. E. Smith, *Authentication: From Passwords to Public Keys*, Addison-Wesley Professional, 2001.

[15] T. Theobald, *On the frontiers of polynomial computations in tropical geometry*, J. Symbolic Comput. **41** (2006), 1360–1375.

CNRS, Mathématiques, Université de Lille, 59655, Villeneuve d'Ascq, France
Email address: `dmitry.grigoryev@math.univ-lille1.fr`

Department of Mathematics and Statistics, Texas Tech University, Lubbock, TX 79409, USA
Email address: `c.monico@ttu.edu`

Department of Mathematics, The City College of New York, New York, NY 10031, USA
Email address: `shpilrain@yahoo.com`