A Total Break of the Scrap Digital Signature Scheme

Daniel Smith-Tone[1,2]

[1] University of Louisville, Louisville KY, USA
[2] National Institute of Standards and Technology, Gaithersburg, Maryland, USA
daniel.smith@nist.gov

Abstract. Recently a completely new post-quantum digital signature scheme was proposed using the so called "scrap automorphisms". The structure is inherently multivariate, but differs significantly from most of the multivariate literature in that it relies on sparsity and rings containing zero divisors. In this article, we derive a complete and total break of Scrap, performing a key recovery in not much more time than verifying a signature. We also generalize the result, breaking unrealistic instances of the scheme for which there is no particularly efficient signing algorithm and key sizes are unmanageable.

Keywords: Multivariate Cryptography · Scrap · Cryptanalysis · Gröbner basis.

1 Introduction

In recent years, we have seen a great deal of development of new proposals for post-quantum cryptosystems based on completely different problems than the majority of secure schemes throughout the history of post-quantum cryptography. Sometimes these new ideas seem to lead to a completely new direction for cryptography. For example, the Picnic digital signature scheme, see [6], which notably advanced quite far in the standardization process for post-quantum schemes of the National Institute of Standards and Technology (NIST), see [4], inspired many of the now called "MPC-in-the-head" class of schemes.

Still, 21 of the 69 original proposals to NIST's process were broken in the first round, see [5], and offer a warning that very different proposals are necessarily relatively untested. Thus, while it is valuable for science to see many new proposals, we build confidence in security based on meaningful security arguments and, rather more importantly, lengthy analysis periods in which a scheme is under constant scrutiny. This time $\times$ attention metric is precisely what is provided to proposals under processes such as NIST's first call for proposals on post-quantum cryptographic algorithms and its additional call for post-quantum digital signatures.

This article is closely related to the latter of the above two mentioned phenomena. Specifically, this article provides a complete break of the new digital signature scheme based on "scrap automorphisms" of [3].

The digital signature scheme of [3] introduces a novel approach to constructing an efficient trapdoor function based on rectangular matrices with coefficients in a polynomial ring. These matrices must satisfy a particular notion of smallness for efficiency in terms of signature size and public key size as well as signing and verification time. We demonstrate that this smallness property necessarily makes the scheme susceptible to algebraic methods of attack.

In addition to its relevance in breaking the "Scrap" digital signature scheme, our method is instructive in how one may develop multivariate style cryptosystems. Specifically, our algebraic cryptanalysis does not necessarily depend on finding a Gröbner basis. We are able to break any instance of the scheme by restricting the degrees of the polynomials in our generating set. The same cryptanalysis should break any scheme revealing relatively "small" secret multiples of a known ideal basis.

The paper is organized as follows. First, we present the novel approach to constructing a trapdoor map given by the Scrap digital signature scheme. Next, we present our cryptanalysis and explain why it devastates Scrap. We then present our experiments performing the attack for various parameter sets. Finally, we conclude, reviewing what we have learned from this experience.

2 The Scrap Digital Signature Scheme

The Scrap digital signature scheme was proposed in [3]. Their construction relies on what they call "scrap" automorphisms.

Let $q > 1$ be a positive (possibly composite) integer and consider the ring $R = \mathbf{Z}_q[x_1, x_2, \ldots, x_n]$ for some positive integer n. Select positive integers $\ell < k$. The Scrap digital signature private and public keys are given by $\mathbf{S} \in R^{\ell \times k}$ and $\mathbf{P} \in R^{k \times \ell}$ with the property that $\mathbf{SP} = \mathbf{I}_\ell$, the $\ell \times \ell$ identity matrix.

To sign a message m, first one computes a hash value $H(m)$ and encodes this value into a vector $\mathbf{u} = \mathrm{Enc}(H(m))$ of ℓ polynomials in R. Then using the private key, the signer generates $\mathbf{v} = \mathbf{uS}$. The signature is then $\mathbf{v}$.

To verify a signature $\mathbf{v}$ for a message m, the verifier first repeats the public process of hashing the message and encoding it into the vector $\mathbf{u} = \mathrm{Enc}(H(m))$. Then it is checked that $\mathbf{u} = \mathbf{vP}$. Correctness is ensured due to the fact that $\mathbf{vP} = \mathbf{uSP} = \mathbf{uI}_\ell = \mathbf{u}$.

Key generation is accomplished by constructing an invertible matrix $\mathbf{A}$, and setting $\mathbf{P}$ to be a collection of ℓ of the columns of $\mathbf{A}$ while $\mathbf{S}$ is selected to be the corresponding rows of $\mathbf{A}^{-1}$.

Naturally, without any constraints, the size of the elements in the coordinate of the above products can grow to completely impractical sizes. To combat this explosion in size and complexity, Scrap uses private and public matrices consisting of t-sparse polynomials with degree bound b, where t and b are additional parameters of the system. We leave the details on the encoding function Enc and the responsible selection of random degree bound b polynomials that are t-sparse to [3, Section 3].

In fact, the most straightforward way of constructing an inverse for a matrix in $R^{k\times k}$ is to compute the adjugate matrix and multiply by the inverse of the determinant, which we must assume is a unit in R. (It is easy to show for the case of $q = 6$, used in the Scrap specification, that the unit group of R is $\{1, 5\}$, the cyclic group generated multiplicatively by the element $5 \in R$.) In general, this process produces higher degree and less sparse polynomials for sparse matrices.

Thus, due to the claims in [3] of the same size for the private key as the public key, it is apparent that the complete details of the method for generating keys satisfying the sparsity and degree bound are not provided in [3]. Via private communication with the authors, we discovered that their complete method for constructing private key/public key pairs involves computing the matrix $\mathbf{A}$ above as a product of a short list of elementary matrices, so that $\mathbf{A}^{-1}$ can be computed as a product of elementary matrices of the same length. Since any invertible matrix can be written as a product of elementary matrices, it is necessarily the case that these matrices are selected from a strange distribution.

Since for any matrix of the form $\mathbf{A} = \mathbf{E}_1 \cdots \mathbf{E}_r$ we have that $\mathbf{A}^{-1} = \mathbf{E}_r^{-1} \cdots \mathbf{E}_1^{-1}$, and due to the fact that the order of the product of elementary matrices makes a difference in the sparsity of the polynomials in the product, the task of choosing the $\mathbf{E}_i$ to make $\mathbf{A}$ sparse typically makes $\mathbf{A}^{-1}$ denser. Thus, the best strategy for producing private and public keys of the same storage size is to generate $\mathbf{A}$ as the product of lower and upper unitriangular matrices $\mathbf{L}$ and $\mathbf{U}$, where either $\mathbf{L}$ and $\mathbf{U}^{-1}$ are chosen to be sparser while $\mathbf{L}^{-1}$ and $\mathbf{U}$ are denser, or vice versa. Then the average sparsity of the coefficients of $\mathbf{A}$ is roughly the same as $\mathbf{A}^{-1}$ and can be controlled.

3 The Algebraic Attack

The Scrap digital signature scheme presented in Section 2 is completely insecure. In this section, we present a general attack structure that breaks the specification of Scrap as well as other possible parameterizations that are less efficient in signature size and private key sizes.

First, note that the "scrap" automorphisms $\phi : R^\ell \to R^k$ of [3] defined by $\phi(\mathbf{u}) = \mathbf{uS}$, where $\mathbf{S} \in R^{\ell\times k}$ has a right inverse, are, indeed, isomorphisms from R^ℓ to R-submodules of R^k given by the image of the maps.

Lemma 1 *The map $\phi : R^\ell \to R^k$ given by $\mathbf{u} \overset{\phi}{\mapsto} \mathbf{uS}$, where $\mathbf{S}$ has a right inverse $\mathbf{P}$ is an isomorphism of R-modules between R^ℓ and $\phi(R^\ell)$.*

Proof. This lemma is an immediate consequence of the commutativity of scalar multiplication with matrix multiplication, the distributivity of matrix multiplication over addition, and the fact that the inverse map is also homomorphic in the same way.

More significantly, note that the ideal generated by the coordinates of a vector $\mathbf{u}$ in R^ℓ is the same ideal in R as that generated by $\phi(\mathbf{u}) = \mathbf{uS}$.

Lemma 2 *Given* $\mathbf{u} \in R^\ell$ *and a scrap automorphism* ϕ*, we have that the ideal generated by the coordinates of* $\mathbf{u}$ *is equal to the ideal generated by* $\phi(\mathbf{u})$*.*

Proof. Note that for any vector $\mathbf{w} \in R^k$, we have that

$$\mathbf{u}\mathbf{S}\mathbf{w}^\top = \mathbf{u}\left(\mathbf{S}\mathbf{w}^\top\right),$$

by the associativity of matrix multiplication. Therefore, any R-combination of the coordinates of $\mathbf{uS}$ is an R-combination of the coordinates of $\mathbf{u}$, which is to say that the ideal generated by the coordinates of $\mathbf{uS}$ is contained in the ideal generated by the coordinates of $\mathbf{u}$.

Similarly, for any vector $\mathbf{w} \in R^\ell$, we have that

$$\mathbf{u}\mathbf{w}^\top = \mathbf{u}\mathbf{S}\mathbf{P}\mathbf{w}^\top = (\mathbf{u}\mathbf{S})\left(\mathbf{P}\mathbf{w}^\top\right),$$

where $\mathbf{SP} = \mathbf{I}_\ell$. Therefore, the ideal generated by the coordinates of $\mathbf{u}$ is contained in the ideal generated by the coordinates of $\mathbf{uS}$. Thus, since we have containment in both directions, the ideals are equal.

Since the coordinates of both the vector $\mathbf{u} = \text{Enc}(H(m))$ and $\mathbf{v} = \mathbf{uS}$ are generating sets for the same ideal, each can be generated from the other. Indeed, the public key $\mathbf{P}$ provides the R-combination of coordinates of $\mathbf{v}$ producing $\mathbf{u}$. In the other direction, the secret key provides R-combinations of the coordinates of $\mathbf{u}$ producing v. A step in key recovery, then, is to find R-combinations of the coordinates of $\mathbf{u}$ producing the coordinates of $\mathbf{v}$.

In general, one may compute a Gröbner basis of the ideal generated by $G_0 = \{u_1, \ldots, u_\ell\}$, the coordinates of $\mathbf{u}$. If a composite integer q is selected, this calculation can become convoluted; however, it is always possible to construct a small degree bounded generating set in stages by either following a Buchberger strategy: adding S-polynomials of generators to the set and reducing modulo the remaining polynomials, or following an F4 strategy: multiplying by all degree 1 monomials and reducing modulo the remaining polynomials. With either strategy, one may produce low degree generating sets $G_1, G_2, \ldots$ in stages.

Given a "nice" generating set G of $I = \langle u_1, \ldots, u_\ell \rangle$ consisting of small degree polynomials, we may attempt to recover an R-combination of the generators producing each coordinate of $\mathbf{v}$ by attempting to solve the computational ideal membership problem with the basis G. This task is performed by reducing the coordinates of $\mathbf{v}$ by generators in this set to determine if it is possible to reduce to 0. The R-combinations produced in this fashion, then, are candidates for the columns of an equivalent secret key $\mathbf{S}'$. Finally, the candidate can be tested as an equivalent secret key by computing $\mathbf{S}'\mathbf{P}$ to determine if it is $\mathbf{I}_\ell$.

Since the secret key $\mathbf{S}$ has coordinates that are relatively sparse polynomials, we see that $\mathbf{v}$ has an extra property: every coordinate is a sparse combination of the coordinates of $\mathbf{u}$. Thus, we merely must recover sparse R-combinations of $u_1, \ldots, u_\ell$ producing the coordinates v_i of $\mathbf{v}$.

Given n variables and the fact that we have that

$$\begin{aligned} v_i &= \sum_{j=1}^{\ell} s_{j,i} u_j \\ &= \sum_{j=1}^{\ell} \sum_{r=1}^{t} m_{j,i,r} u_j, \end{aligned} \tag{1}$$

where $m_{j,i,r}$ is a degree bound b monomial, the probability that there is cancellation between any two summands above for large n and small t and b is very low, though it depends on the distributions used for the random selection of the $m_{j,i,r}$ in key generation.

To turn this observation into an attack, we examine the coordinates v_i of $\mathbf{v}$ and record all coordinates u_j whose leading terms divide the leading term of v_i. Computing the quotient and remainder upon division by some choice of u_j, we may repeat this step for the remainder. Considering all possible choices for u_j at each step, we obtain a search tree for recovering an R-combination of the u_j producing v_i. This search tree represents all possible orders of divisors of v_i by the u_j; thus, if v_i is generated by the u_j without any reductions of leading terms, the algorithm will terminate with a correct R-combination.

Thus, the most straightforward attack is to do a depth-first search on this tree structure, recovering an expression $v_i = \sum_{j=1}^{\ell} s'_{j,i} u_j$. Combining all such polynomials $s'_{j,i}$ into the matrix $\mathbf{S}'$ produces our candidate secret key that can be validated via multiplication with the public key.

If n is not too large in comparison to t and b, then the probability of cancellation of terms in (1) becomes higher, and, therefore, the probability that v_i does not reduce to zero in the search tree increases. To remedy this situation, we may increase the size of the generating set by adding S-polynomials and reducing (i.e., we can transition from using the generating set G_i to using G_{i+1} as described above), and then starting the calculation again with a higher probability of a reduction to zero in the search tree.

An outline of the full attack with the size of generating set as a parameter is given in Algorithm 1. The algorithm calls on two subroutines, Buchberger and IdealMembership. The Buchberger subroutine uses a parameter i indicating the depth of S-polynomials included in an application of Buchberger's Algorithm, see [2]. For example, if $i = 0$, then the input sequence is top-reduced with respect to itself and returned. If $i = 1$, then all S-polynomials are added and the sequence is top-reduced with respect to itself and returned. The IdealMembership routine performs a depth-first search on the order of top divisors of the input polynomial with respect to the input basis to recover a sequence of R-coefficients verifying membership in the ideal. To simplify the presentation and avoid pointers in pseudocode, the algorithm is presented in Algorithm 2 executing multiple returns on different threads with failures marked with $\perp$.

Algorithm 1 breaks Scrap with certainty for a sufficiently large generating set G incorporating sufficiently many reductions of leading terms. The complexity of the attack is then determined by the size of G and the average degree of the

Algorithm 1 KeyRecovery

Input: $\mathbf{u} = (u_1, \ldots, u_\ell) \in R^\ell$, $\mathbf{v} \in R^k$ and i.
Output: $\mathbf{S}'$ st $\mathbf{S}'\mathbf{P} = \mathbf{I}_\ell$ or some $s_j = \perp$ if no such $\mathbf{S}'$ exists.

$G_i \leftarrow$ Buchberger$(\{u_1, \ldots, u_\ell\}, i)$
for j **from** 1 **to** k **do**
  $\mathbf{s}_j \leftarrow$ IdealMembership(G_i, v_j)
end for
return $\mathbf{S}' = \left[\mathbf{s}_1^\top \cdots \mathbf{s}_k^\top\right]$

Algorithm 2 IdealMembership

Input: $G = \{g_1, \ldots, g_m\} \subset R$, $f \in R$.
Output: $\mathbf{s} \in R^m$ st $\sum_i s_i g_i = f$ or $\perp$ if no such $\mathbf{s}$ exists.

if there exists $g_i \in G$ **st** LeadingTerm(g_i) **divides** LeadingTerm(f) **then**
  for all $g_i \in G$ **st** LeadingTerm(g_i) **divides** LeadingTerm(f) **do**
    $\mathbf{s} \leftarrow \perp$
    $c \leftarrow$ LeadingTerm(f)/LeadingTerm(g_i)
    $f' \leftarrow f - cg_i$
    $\mathbf{s} \leftarrow$ IdealMembership(G, f')
    if $\mathbf{s} \neq \perp$ **then**
      $s_i \leftarrow s_i + c$
    end if
    return $\mathbf{s}$
  end for
else
  return $\perp$
end if

search tree. Again, when n is sufficiently large in comparison to t and b, it is unlikely that the depth of reduction in the required generating set is very high, due to the general sparsity of the terms in the Macaulay matrix. Thus, it is very specifically the required sparsity of the secret key that permits this attack.

4 Experiments and Complexity Analysis

We implemented the attack from the previous section in the Magma Computer Algebra System[3], see [1]. We performed these experiments on the full size parameters as suggested in [3] as well as small toy variants in which the secret key is not as sparse and the public key is a random left-invertible matrix.

For the real parameters, we note that in all instances the attack succeeded, taking the generating set for ideal membership to be merely the $\mathbf{Z}_q$ ($q = 6$)

[3] Certain equipment, instruments, software, or materials, commercial or non-commercial, are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement of any product or service by NIST, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

multiples of coordinates of $\mathbf{u}$, i.e., the parameter i in Algorithm 1 is 0, indicating that n was sufficiently large for cancellation in Equation (1) to be a low probability event. We were able to recover the same secret key as the legitimate user in all cases in milliseconds. The results are summarized in Table 1.

Table 1. MAGMA attack timing for 1000 instances of the Scrap digital signature scheme with smaller sparsity bound t and for claimed NIST Security level I, i.e., 143-bit security, parameters.

Scrap(q, n, ℓ, k, t, b)	Least(ms)	Average(ms)	Most(ms)
$Scrap(6, 64, 5, 10, 2, 3)$	20	100	320
$Scrap(6, 64, 5, 10, 3, 3)$	30	1140	4170

To study the difference between the scheme as published in [3] and the actual scheme incorporating elementary matrices to produce the secret and public matrices, we also implemented the scheme with a uniformly random t-sparse left invertible public key. The main difference in this scheme is that the corresponding secret key is significantly dense. Still, our attack was able to break these instances, even using the generating set $G_0 = u_1, \ldots, u_\ell$ when n was reasonably large. Of course, we were unable to make instances that were very large due to the increased storage size and calculation time for the left-inverse of the public key. We limited our experiments to the cases of $\ell = 3$ and $k = 6$ to study these instances.

For instances with such an unrealistic balance of parameters, we found that the attack still worked in all cases. As an example of using larger generating sets, we fixed $t = 2$ and $b = 3$ to determine what values of n required larger generating sets incorporating S-polynomials. We noted that there appears to be a cut-off phenomenon for such instances between $n = 12$ and $n = 16$. For $n = 16$ every experiment solved the system with the original generating set G_0. For $n = 12$, approximately two thirds of our instances required the generating set G_1 to succeed. While it may be a legitimate mitigation measure to increase t and b relative to n to achieve security, it is completely unrealistic to have secret keys so large; furthermore, signing quickly becomes unreasonable.

5 Conclusion

The idea for Scrap is very clever and inviting. The reality of the situation is, however, not so positive for Scrap. The attack we have derived is quite effective against even extremely unrealistic parameters. Thus, we must conclude that the Scrap digital signature, without some major overhaul, is quite insecure and unusable.

Still, there is an important lesson to learn and value in the science that Scrap helps to advance. This is the first attack the author is aware of in which there is a meaningful difference between computing a Gröbner basis for an ideal and studying the ideal membership problem from other less structured bases.

Scrap teaches us that instances of multivariate schemes for which there is a relationship among public data that is carried by a sparse secret may exhibit weakness against algebraic attack. This statement may seem obvious after this paper's discussion; however, much of post-quantum cryptography is related to a secret that is sparse in some sense. Multivariate schemes often rely on low degrees or restricted monomials. Code-based schemes often rely on an error that is small in some metric, as do lattice-based cryptosystems. The issue of which metric allows sparse secrets to remain safe is a very deep topic that is subtly suggested by the direction the Scrap designers chose. We all still have a lot to learn.

References

1. Bosma, W., Cannon, J., Playoust, C.: The magma algebra system i: The user language. J. Symb. Comput. **24**(3–4), 235–265 (Oct 1997). https://doi.org/10.1006/jsco.1996.0125, https://doi.org/10.1006/jsco.1996.0125
2. Buchberger, B.: A theoretical basis for the reduction of polynomials to canonical forms. SIGSAM Bull. **10**(3), 19–29 (aug 1976). https://doi.org/10.1145/1088216.1088219, https://doi.org/10.1145/1088216.1088219
3. Chen, J., Grigoriev, D., Shpilrain, V.: Digital signature schemes using non-square matrices or scrap automorphisms. CoRR **abs/2306.08927** (2023). https://doi.org/10.48550/arXiv.2306.08927, https://doi.org/10.48550/arXiv.2306.08927
4. Group, C.T.: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. NIST CSRC (2016), http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/call-for-proposals-final-dec-2016.pdf
5. Moody, D.: Presentation: The second round of the nist pqc standardization process. NIST First Post-Quantum Standardization Conference, NIST CSRC (2018), https://csrc.nist.gov/CSRC/media/Presentations/the-2nd-round-of-the-nist-pqc-standardization-proc/images-media/moody-opening-remarks.pdf
6. Zaverucha, G., Chase, M., Derler, D., Goldfeder, S., Kales, D., Katz, J., Kolesnikov, V., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Wang, X.: Picnic. NIST CSRC (2020), https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions

A Toy Example

We illustrate the attack by attacking a toy instance of Scrap with smaller parameters. We choose: $n = 16$, $\ell = 3$, $k = 6$, $t = 2$ and $b = 3$.

A.1 Key Generation

We generate upper and lower unitriangular matrices $\mathbf{U}$ and $\mathbf{L}$ by using elementary matrices $E_{i,j}$ and $F_{i,j}$, respectively. Each matrix is defined by its single

off-diagonal nonzero entry, $e_{i,j}$ or $f_{i,j}$ in R. A list of these values for these matrices is the following:

$$\begin{aligned}
e_{5,6} &= 4x_5 + 4x_7 \\
e_{4,5} &= 4x_2x_{13}x_{14} + 2x_3x_9 \\
e_{3,4} &= 5x_1x_9 + x_5x_9 \\
e_{2,3} &= 2x_{14}x_{16} + 5x_4 \\
e_{1,2} &= x_3x_6x_{13} + x_4x_{15} \\
f_{6,5} &= 3x_3x_9 + 3x_{10} \\
f_{5,4} &= 5x_2x_5x_{10} + 3x_{13}x_{14} \\
f_{4,3} &= 3x_1 + x_{16} \\
f_{3,2} &= 5x_2 \\
f_{2,1} &= 3x_6^2x_{11} + 2x_7x_{11}x_{15}.
\end{aligned}$$

Then we recover $\mathbf{L}$ and $\mathbf{U}$ from these elementary matrices using the above order. We then choose a random permutation of the columns to make a random selection of the invertible matrix $\mathbf{A}' = \mathbf{UL}$. Using the permutation matrix

$$\mathbf{Q} = \begin{bmatrix} 0&0&0&0&1&0 \\ 1&0&0&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&1&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&0&0&0&1 \end{bmatrix},$$

we construct $\mathbf{A} = \mathbf{ULQ}$ and $\mathbf{A}^{-1} = \mathbf{Q}^{-1}\mathbf{L}^{-1}\mathbf{U}^{-1}$. Then the public key $\mathbf{P}$ consists of the first $\ell = 3$ columns of $\mathbf{A}$ and the secret key $\mathbf{S}$ consists of the first 3 rows of $\mathbf{A}^{-1}$.

A.2 A Signature

We suppose that the $\mathbf{u} = \mathrm{Enc}(H(m))$ is given by

$$\mathbf{u}^\top = \begin{bmatrix} x_2^2x_3^2x_6x_7x_{11}x_{12}x_{14} + 4x_2x_4^2x_5x_6x_{10}x_{13} + 2x_2x_6x_{10} + 2x_{10} \\ 4x_1^2x_4x_5x_8^2x_9x_{10} + 5x_6x_7x_9x_{12}^3x_{14}x_{16} + 5x_1x_4x_5x_{11}x_{16} + x_5x_8x_9x_{12} \\ x_2x_6x_8x_9^2x_{13}^2 + 3x_3x_8x_9x_{13}x_{14} + 4x_4 + 3x_5 \end{bmatrix}.$$

The signature is then given by $\mathbf{v} = \mathbf{uP}$.

A.3 Key Recovery

We go in detail into the recovery of the first column of the secret key $\mathbf{S}$. The first column of $\mathbf{S}$ only affects the first coordinate of the signature $\mathbf{v}$, so we only need to consider

$$\begin{aligned}
v_1 =& 3x_2^2x_3^2x_6^3x_7x_{11}^2x_{12}x_{14} + 4x_2^2x_3^2x_6x_7^2x_{11}^2x_{12}x_{14}x_{15} + 4x_2x_4^2x_5x_6x_7x_{10}x_{11}x_{13}x_{15} \\
&+ 2x_2x_6x_7x_{10}x_{11}x_{15} + 2x_7x_{10}x_{11}x_{15}
\end{aligned}$$

in deriving the coefficients $s_{i,1}$ for $1 \leq i \leq \ell$.

Using the coordinates of $\mathbf{u}$ as a generating set, we see that the leading term of v_1 is divisible by the leading term of u_1. In particular, we compute

$$c_1 = \mathrm{LT}(v_1)/\mathrm{LT}(u_1) = 3x_6^2 x_{11},$$

and use this value to compute the value $f_2 = v_1 - c_1 u_1$ producing the value

$$\begin{aligned} f_2 =& 4x_2^2 x_3^2 x_6 x_7^2 x_{11}^2 x_{12} x_{14} x_{15} + 4x_2 x_4^2 x_5 x_6 x_7 x_{10} x_{11} x_{13} x_{15} \\ &+ 2x_2 x_6 x_7 x_{10} x_{11} x_{15} + 2x_7 x_{10} x_{11} x_{15}. \end{aligned}$$

Continuing, we notice that the leading term of f_2 is divisible by u_1. Specifically, we obtain

$$c_2 = \mathrm{LT}(f_2)/\mathrm{LT}(u_1) = 4x_7 x_{11} x_{15}.$$

Reducing the remainder f_2 again, we find that $f_3 = f_2 - c_2 u_1 = 0$. Thus, we have that

$$v_1 = \left(3x_6^2 x_{11} + 4x_7 x_{11} x_{15}\right) u_1 + 0u_2 + 0u_3,$$

and we have recovered the first column of $\mathbf{S}$,

$$\mathbf{s}_1^\top = \left[3x_6^2 x_{11} + 4x_7 x_{11} x_{15}\ 0\ 0\right]^\top.$$

The remaining columns are recovered in the same manner, reproducing the private key $\mathbf{S}$ exactly.