Short Solutions to Nonlinear Systems of Equations

Alan Szepieniec and Bart Preneel

imec-COSIC KU Leuven, Belgium
first-name.last-name@esat.kuleuven.be

Abstract. This paper presents a new hard problem for use in cryptography, called Short Solutions to Nonlinear Equations (SSNE). This problem generalizes the Multivariate Quadratic (MQ) problem by requiring the solution be short; as well as the Short Integer Solutions (SIS) problem by requiring the underlying system of equations be nonlinear. The joint requirement causes common solving strategies such as lattice reduction or Gröbner basis algorithms to fail, and as a result SSNE admits shorter representations of equally hard problems. We show that SSNE can be used as the basis for a provably secure hash function. Despite failing to find public key cryptosystems relying on SSNE, we remain hopeful about that possibility.

Keywords: signature scheme, hard problem, post-quantum, MQ, SIS, SSNE, hash function

1 Introduction

The widely deployed RSA and elliptic curve cryptosystems rely on the hardness of the integer factorization and discrete logarithm problems respectively, which are in fact easy to solve on quantum computers by means of Shor's algorithm [31]. These encryption and signature schemes will therefore become insecure once large enough quantum computers are built; and as a result we need to design, develop and deploy cryptography capable of resisting attacks by quantum computers, despite running on classical computers.

A number of hard problems have been proposed to replace integer factorization and discrete logarithms for precisely this purpose of offering *post-quantum* security. For instance, the problem of finding short vectors in high-dimensional lattices relates to normed linear algebra problems such as SIS [1] and LWE [29], which in turn generate many types of public key cryptosystems. Finding satisfying solutions to systems of multivariate quadratic (MQ) systems of equations seems to be hard even if the quadratic map embeds a secret trapdoor allowing only the secret-key holder to generate signatures [14]. Evaluating isogenies between elliptic curves is easy, but finding the isogeny from input and output images is hard; this enables a rather direct adaptation of the Diffie-Hellman key agreement protocol [20]. Even traditionally symmetric problems such as

hash function inversion have been used to generate stateless digital signature schemes [5]. However, in nearly all post-quantum cryptosystems to date, either the public key or else the ciphertext or signature is huge — measurable in tens of kilobytes if not megabytes[1]. In the interest of easing the transition away from the quantum-insecure but very low-bandwidth ECDSA, designing a post-quantum signature scheme with short signatures or ciphertexts *and* short public keys is a major open problem.

In this paper, we propose a new cryptographic problem called Short Solutions to Nonlinear Equations (SSNE) and argue that it is likely hard, even for quantum computers. Informally, our new hard problem asks to find a *short* solution to a system of *non-linear* multivariate polynomial equations, and thus generalizes both the Short Integer Solution (SIS) problem where the system is linear, and the Multivariate Quadratic (MQ) problem where the solution need not be short. Adopting both requirements renders standard attack strategies inapplicable or wildly inefficient.

Nevertheless, we show in Section 4 that it is possible to attack SSNE with limited success, in a way that improves over brute force search. We take this attack and its limitations into account and delineate a niche of parameter space in which brute force is the most efficient attack strategy. As a result, SSNE offers a denser encoding of computational hardness than either SIS or MQ, and if it is possible to design public key cryptosystems that rely on this hard problem, it holds promise of generating a smaller public keys, ciphertexts and signatures than their MQ and SIS counterparts without incurring a security cost.

While designing a public key cryptosystem on top of SSNE remains an open problem, designing a hash function whose security relies on SSNE does not, as this problem is solved in Section 5. This result does not merely serve to demonstrate design of cryptographic primitives in lieu of the comparably more difficult end-goal of designing public key functionalities; it has standalone value as well. From the point of view of provable security, very few hash functions come with a security proof showing that finding a solution implies solving a hard problem that is defined independently of the hash function itself. Therefore these not-provably-secure hash functions offer less assurance of security than provably secure hash functions whose underlying hard problems are studied independently. Moreover, it is prudent to diversify the hard problems upon which cryptographic primitives rely, in order to isolate the effects of cryptanalytic breakthroughs.

2 Preliminaries

Notation. We denote by $\mathbb{F}_q$ the finite field of q elements. The integer range $\{a, a+1, \ldots, b-1, b\}$ is denoted by $[a : b]$. Vectors are denoted in boldface, *e.g.*, $\mathbf{x}$ and matrices by capital letters, *e.g.*, A, with indexation starting at zero. The

[1] The curious exception to this rule is the supersingular isogeny Diffie-Hellman key agreement scheme, but even so it does not seem possible to use this construction for small signature schemes.

slice of A consisting of rows i—j and columns k—l is denoted by $A_{[i:j,k:l]}$, and we drop the $, k:l$ when slicing from a vector instead of a matrix.

Lattices. A *lattice* of dimension n and embedding degree m is a discrete n-dimensional subspace of $\mathbb{R}^m$; without loss of generality, we consider subspaces of $\mathbb{Z}^m$. Any such lattice $\mathcal{L}$ can be described as the set of integer combinations of a set of vectors $\mathbf{b_0}, \ldots, \mathbf{b_{n-1}} \in \mathbb{Z}^m$, which is called a *basis* for the lattice and is not unique for a given lattice. A lattice $\mathcal{L}$ is *q-ary* whenever membership of a point $\mathbf{p} \in \mathbb{Z}^m$ is decided by $\mathbf{p} \bmod q$, *i.e.*, with each component reduced modulo q.

The LLL algorithm [24] takes a matrix of integers $A \in \mathbb{Z}^{h \times w}$ whose rows span a lattice, and outputs another matrix $B \in \mathbb{Z}^{h \times w}$ whose rows span the same lattice but are much shorter in length. Without loss of generality we assume the LLL algorithm also outputs a unitary matrix U such that $UA = B$. The shortest basis vector produced by LLL when applied to a lattice spanned by h vectors of w elements, is bounded in length by

$$\|\mathbf{b_0}\|_2 \leq \left(\frac{4}{4\delta - 1}\right)^{(w-1)/4} \mathsf{det}(\mathcal{L})^{1/w} \ , \tag{1}$$

where $\frac{1}{4} < \delta \leq 1$ is the LLL parameter and where the *determinant of the lattice* is given by $\mathsf{det}(\mathcal{L}) = \mathsf{det}(AA^\mathsf{T})^{1/2} = \mathsf{det}(BB^\mathsf{T})^{1/2}$ if A and B have linearly independent rows.

In the case of q-ary matrices, a basis matrix can be obtained by adjoining the original basis matrix with $q\mathrm{I}$. LLL will return a $(w+h) \times w$ matrix whose first w rows consist of all zeros. The determinant of q-ary lattices of this dimension is q^{w-h} with high probability [27], which means that the length of the shortest nonzero vector in the output of LLL is bounded by

$$\|\mathbf{b_0}\|_2 \leq \left(\frac{4}{4\delta - 1}\right)^{(w-1)/4} q^{(w-h)/w} \ . \tag{2}$$

The ith *successive minimum* $\lambda_i(\mathcal{L})$ of a lattice $\mathcal{L}$ is the smallest $\rho \in \mathbb{R}$ such that the hypersphere with radius ρ centered at the origin contains at least i independent lattice points. According to the m-dimensional ball argument of Micciancio and Regev [27], the first successive minimum of a random q-ary lattice of dimension h and embedding dimension w can be approximated by

$$\lambda_0(\mathcal{L}) \approx \sqrt{\frac{w}{2\pi e}}\, q^{(w-h)/w} \ . \tag{3}$$

3 Short Solutions to Nonlinear Equations

Our hard problem generalizes the Multivariate Quadratic (MQ) problem as well as the Short Integer Solution (SIS) problem. After presenting the definitions we

consider some straightforward attacks. In the next section we consider a more sophisticated one.

MQ Problem. Given a quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ consisting of m polynomials in n variables of degree at most 2, find a vector $\mathbf{x} \in \mathbb{F}_q^n$ such that $\mathcal{P}(\mathbf{x}) = \mathbf{0}$.

The MQ problem is **NP**-hard in general as well as empirically hard on average whenever $m \approx n$. The best known attack is the hybrid attack [6], which consists of guessing some variables so as to overdetermine the system of equations and then solving it using a Gröbner basis type solver such as F_4 [16] or XL [13]. The reduced cost of solving the overdetermined system compensates for the increased cost of retrying a new guess whenever it leads to no solutions. The complexity of the optimal-trade-off hybrid attack approaches $2^{C_q n}$ as $n \gg q \to \infty$ with $C_q = \omega(1.38 - 0.44\,\omega \log_2 q)$ and where $\omega \geq 2$ is the exponent of matrix multiplication complexity [7]. However, when $q \gg n$, the cost of even one random guess beyond the number of variable-fixes that makes the system a determined one, dominates the attack complexity. In this case the complexity of a purely algebraic attack can be estimated using the *degree of regularity* D_{reg} of the system. For semi-regular quadratic systems [4,3] (which we assume random quadratic systems are), the degree of regularity is equal to the degree of the first term with a non-positive coefficient of the power series expansion of

$$\text{HS}(z) = \frac{(1-z^2)^m}{(1-z)^n} \quad . \tag{4}$$

At this point, the Gröbner basis computation using F_4 or XL boils down to performing sparse linear algebra in the Macaulay matrix whose polynomials have degree D_{reg}. The complexity of this task is $O\left(\binom{n+D_{\text{reg}}+1}{D_{\text{reg}}}^2\right)$ in terms of the number of finite field operations, which in turn are polynomial in $\log q$. In summary, the complexity of solving the MQ problem is *exponential* in $n \approx m$, but barely affected by q.

SIS Problem. Given a matrix $A \in \mathbb{F}_q^{n \times m}$ with $m > n$, find a nonzero vector $\mathbf{x} \in \mathbb{Z}^m \backslash \{\mathbf{0}\}$ such that $A\mathbf{x} = \mathbf{0} \bmod q$ and $\|\mathbf{x}\|_2 \leq \beta$.

While not **NP**-hard, SIS does offer an average-case to worst-case reduction: solving random SIS instances is at least as hard as solving the lattice-based Shortest Independent Vectors Problem (SIVP) up to an approximation factor of $\tilde{O}(\beta\sqrt{n})$ in the worst case [26]. The most performant attack on SIS is indeed running a lattice-reduction algorithm such as BKZ 2.0 [8] to find short vectors in the associated lattice which is spanned by the kernel vectors of A. The complexity of this task is captured by the *root Hermite factor* $\delta > 1$, which approaches 1 for more infeasible computations. For a given δ the optimal number of columns of A to take into account (*i.e.*, by setting the coefficients of $\mathbf{x}$ associated to the other columns to zero) is given by $m = \sqrt{n \log_2 q / \log_2 \delta}$. At this point the average length of the lattice points found is $2^{2\sqrt{n \log_2 q \log_2 \delta}}$ and cryptographic

security requires β to be smaller than this number. Albrecht *et al.* estimate the complexity of obtaining lattice points of this quality as $0.009/\mathsf{log}_2^2\delta + 4.1$ in terms of the base-2 logarithm of the number of time steps [2]. The key takeaway is that the complexity of SIS grows *exponentially* in m and n, but *polynomially* in q and β.

SSNE Problem (Short Solutions to Nonlinear Equations) Given a map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ consisting of m polynomials in n variables over a prime field $\mathbb{F}_q$ and with $\mathsf{deg}(\mathcal{P}) \geq 2$, find a vector $\mathbf{x} \in \mathbb{Z}^n$ such that $\mathcal{P}(\mathbf{x}) = 0 \bmod q$ and $\|\mathbf{x}\|_2 \leq \beta$.

It is clear that the attack strategies that work for MQ and SIS do not apply out of the box to the SSNE problem. The random guess of the hybrid attack on MQ might fix the first few variables to small values, but offers no guarantee that an algebraic solution to the other variables is small. Alternatively, one can drop the random guess and compute a Gröbner basis for the under-determined system. Even if the resulting Gröbner basis consists of a reasonable number of polynomials of reasonable degrees, obtaining a short vector in the variety associated with the Gröbner basis seems like a hard problem in and of itself. Alternatively, one can linearize the system by introducing a new variable for every quadratic term and treat the resulting matrix of coefficients as the matrix of a SIS instance. However, in this case it is unclear how to find the correct length bound β as it now applies to a vector of quadratic monomials. Nevertheless, we now show under which conditions or adaptations an algebraic attack and attack based on lattice reduction are possible.

3.1 Algebraic Attack

The constraint $\|\mathbf{x}\|_2 \leq \beta$ can be formulated algebraically. Assume $\beta < q/2$, and let $b = \lfloor \beta \rfloor$. Then any solution $\mathbf{x}$ to the SSNE problem must consist of coefficients in $[-b : b]$. For any such coefficient x_i, the polynomial $\prod_{j=-b}^{b}(x_i - j)$ must evaluate to zero. Therefore, by appending these polynomials to $\mathcal{P}$, one obtains a less under-determined system and perhaps even a determined one. If that is the case, XL and F_4 will find a short solution; however, the Gröbner basis computation must reach degree $2b$ for the added polynomials to make a difference, and for sufficiently large β even this task is infeasible. It is possible to generalize this strategy so as to require that the sums-of-squares of all subsets of the coefficients of $\mathbf{x}$ are smaller than β. This method cannot work when $\beta > q$, but can be effective when β is small — say, a handful of bits.

Alternatively, it is possible to run down the unsigned bit expansion of every component of $\mathbf{x}$ and introduce a new variable $x_{i,j}$ for each bit and one for each component's sign s_i. This transformation adds n equations of the form $x_i = s_i \sum_{j=0}^{\lceil \mathsf{log}_2 q \rceil} 2^j x_{i,j}$, $n\lceil \mathsf{log}_2 q \rceil$ equations of the form $x_{i,j}(1 - x_{i,j}) = 0$, and n equations of the form $(s_i - 1)(s_i + 1) = 0$. The advantage of having access to this bit expansion is that the constraint $\|x\|_2 \leq \beta$ can now be expressed as $\lceil \mathsf{log}_2 q \rceil$ equations modulo q, *even when* $\beta > q$.

In both cases, the system of equations becomes infeasibly large whenever β grows, which is exactly the intended effect from a design perspective. Phrased in terms of the security parameter κ, we have

Design Principle 1: *β must be large:* $\log_2\beta > \kappa$.

Note that β cannot be larger than $\sqrt{n}(q-1)/2$ because in that case *any* solution vector $\mathbf{x}$ satisfies the shortness criterion, which can therefore be forgotten at no cost in favor of a very fast algebraic solution. In fact, we want a random solution to the system of equations to satisfy $\|\mathbf{x}\|_2 \leq \beta$ with at most a negligible probability. Design principle 2 requires this probability to be at most $2^{-\kappa}$, where κ is the targeted security level.

Design Principle 2: *β must not be too large:* $n\log_2 q \geq \kappa + n\log_2\beta$.

3.2 Lattice Attack

In the relatively small dimensions considered for SSNE, basic lattice reduction algorithms such as LLL [23] manage to find the shortest vector in polynomial time with all but absolute certainty. Moreover, the nonlinear system $\mathcal{P}(\mathbf{x}) = \mathbf{0}$ can always[2] be represented as a linear system $P\bar{\mathbf{x}} = \mathbf{0}$, where P is the Macaulay matrix of $\mathcal{P}$ and $\bar{\mathbf{x}}$ is the vector of all monomials in $\mathbf{x}$ that appear in $\mathcal{P}$. If the solution $\mathbf{x}$ to $\mathcal{P}(\mathbf{x}) = \mathbf{0}$ is short enough, then its expansion into $\bar{\mathbf{x}}$ will also be a solution to $P\bar{\mathbf{x}} = \mathbf{0}$ — and might be found quickly by lattice-reducing any basis for the kernel of P and weighting the columns as necessary.

In fact, the vector $\bar{\mathbf{x}}$ associated with a solution $\mathbf{x}$ to $\mathcal{P}(\mathbf{x}) = \mathbf{0}$ will *always* lie in the kernel of P, although not every kernel vector corresponds to a solution. Since $\bar{\mathbf{x}}$ is necessarily in the lattice spanned by the kernel vectors of P, the only way to hide it from lattice-reduction is to make it long — as long as random lattice vectors taken modulo q. The rationale behind the next design principle is to require that some of the quadratic monomials $\bar{\mathbf{x}}$ are of the order of magnitude of q (possibly after modular reduction).

Design Principle 3: $\mathbf{x}$ *must not be too small:* $\log_2\|\mathbf{x}\|_2^2 \geq \log_2 q$.

A straightforward attack strategy to cope with this design principle is to focus only on those columns of P that correspond to the monomials of degree 1 in $\bar{\mathbf{x}}$. Lattice reduction will then find short kernel vectors for this reduced matrix $\tilde{P}$. The attack runs through linear combinations of these small reduced kernel vectors until it finds a small linear combination $\mathbf{c}$ such that $\mathcal{P}(\mathbf{c}) = \mathbf{0}$. A rigorous argument counts the number of points in this lattice that have the correct length and then computes the proportion of them that solve $\mathcal{P}(\mathbf{x}) = \mathbf{0}$, and infers from this a success probability and hence a running time for the attack. A far simpler but heuristic argument pretends that the nonlinear monomials of $\bar{\mathbf{x}}$ multiply with their matching columns from P and thus generate a *uniformly random*

[2] This assumes that $\mathcal{P}$ has no constant terms, but the same arguments apply with minor modifications even if it does.

offset vector $\mathbf{p}$. The attacker succeeds only when $\mathbf{p} + \tilde{P}\mathbf{x} = \mathbf{0}$, which can be engineered to occur with at most a negligible probability.

Design Principle 4: *The output space must be large enough:* $m\log_2 q \geq \kappa$.

Lattice-reduction has been used in the past to find small solutions to univariate and multivariate polynomial equations, for instance in the context of factoring RSA moduli $n = pq$ where some of the bits of p or q are known. These applications of LLL were first discovered by Coppersmith [10,9], and were then expanded on by Howgrave-Graham [19], Jutla [21], Coron [11,12], and most recently by Ritzenhofen [30]. The common strategy behind all these attacks is to generate clever algebraic combinations of the polynomials but which must be linearly independent. LLL is run either on the resulting system's Macaulay matrix or on its kernel matrix to find either polynomial factors with small coefficients or else short roots. However, this family of methods is only effective when the targeted solution is short enough. In particular, if $X_i \in \mathbb{Z}$ is a bound on x_i, *i.e.*, $|x_i| \leq X_i$, then success is only guaranteed whenever for every term $t \in \mathbb{F}_q[\mathbf{x}]$ of every polynomial of $\mathcal{P}$ (interpreted as $t \in \mathbb{Z}[\mathbf{x}]$)

$$|t(X_1, \ldots, X_n)| < q \ . \tag{5}$$

This success criterion is inconsistent with design principle 3.

3.3 Additional Considerations

Note that the shortness constraint $\|\mathbf{x}\|_2 \leq \beta$ does not have to apply to all variables. Even requiring only $\sqrt{\sum_{i \in S} x_i^2} \leq \beta$ where the sum is taken only over a non-empty subset S of the variables suffices to capture the hardness of the problem. More generally, the problem can be defined with respect to any weight matrix $W \in \mathbb{Z}^{n \times n}$, namely by requiring that $\mathbf{x}^\mathsf{T} W \mathbf{x} \leq \beta^2$. Diagonalization of W leads to a partitioning of the variables into one set which should be short and one set whose length does not matter. Nevertheless, one should be careful to ensure that the number of short variables must be larger than the dimension of the variety. Otherwise the shortness constraint is no constraint at all because it is possible to guess the short variables and subsequently solve for the remaining variables using a Gröbner basis algorithm.

Design Principle 5. *There should be more small variables than the dimension of the variety:* $\mathsf{rank}(W + W^\mathsf{T}) > \mathsf{dim}\, V(\mathcal{P}) = n - m$.

Remark. The concise way to capture "the number of variables that must be small after optimal basis change" is indeed $\mathsf{rank}(W + W^\mathsf{T})$. To see this, observe that $\mathbf{x}^\mathsf{T} W \mathbf{x}$ is a quadratic form and therefore equal to $\mathbf{x}^\mathsf{T}(W + A)\mathbf{x}$ for any skew-symmetric matrix A (*i.e.*, square matrix such that $A^\mathsf{T} = -A$). Up to additions of skew-symmetric matrices and up to constant factors we have $W \equiv W + W^\mathsf{T}$. This latter form is preferred for diagonalization, which finds an invertible basis change S such that makes $S^\mathsf{T}(W + W^\mathsf{T})S$ diagonal. The zeros on this diagonal

indicate the variables whose size is unconstrained. Moreover, the rank of $W+W^\mathsf{T}$ cannot change under left or right multiplication by invertible matrices such as S^T or S.

3.4 Estimating Hardness

The main selling point of the SSNE problem is that neither the algebraic solvers nor lattice-reduction algorithms seem to apply, and as a result of this immunity it admits a much conciser encapsulation of cryptographic hardness. In MQ problems, the hardness derives from the large number of variables and equations n and m, and is largely independent of the field size q. In SIS problems, the hardness derives mostly from the large lattice dimension n, although the field size q and length constraint β are not entirely independent. Since both Gröbner basis and lattice-reduction algorithms do not apply, the hardness of SSNE problems must be much more sensitive to the size of the search space than their MQ and SIS counterparts. In particular, this sensitivity allows designers to achieve the same best attack complexity while shrinking m and n in exchange for a larger q — a trade-off that makes perfect sense because in all cases the representation of a single problem instance is *linear* in $\log_2 q$ and *polynomial* in m and n.

All five design principles, including design principle 6 which will be derived in Section 4, have a limited range of applicability. No known algorithm solves SSNE problems for which all six criteria are met, faster than the following brute force search does. In the most optimistic scenario, no such algorithm exists. We invite the academic community to find attacks on SSNE that outperform this brute force search. In Section 5 we propose a hash function whose security relies on the assumption that either such an algorithm does not exist or that if it does, it does not beat brute force by any significant margin.

A brute force strategy must only search across $\mathbb{F}_q^{n-m}$. Each guess of the first $n-m$ variables is followed by an algebraic solution to the remaining system of m equations in m variables. If m is not too large then the task of finding this solution algebraically is rather fast, and the complexity of this joint task is dominated by $O(q^{n-m})$. In quantum complexity, Grover's algorithm [18] offers the usual square root speed-up of $O(q^{(n-m)/2})$.

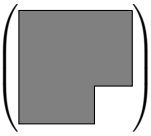
4 An Algebraic-Lattice Hybrid Attack

In this section we describe an attack that applies when $m(m+1)/2 \leq n$ and manages to produce somewhat short solutions. In a nutshell, the attack treats the polynomial system as a UOV$^-$ public key. A UOV reconciliation attack recovers the secret decomposition and at this point the attacker samples vinegar and oil variables such that the resulting "signature" is small. We consider the various steps separately. This section uses the terms "signature" and "solution" interchangeably because in the context of attacks on UOV they are identical.

4.1 UOV

Unbalanced Oil and Vinegar [22] is an MQ signature scheme with parameters $n = o + v$, $v \approx 2o$ and $m = o$. The public key is a homogeneous quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$. The secret key is a decomposition of this public map into $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ and $S \in \mathsf{GL}_n(\mathbb{F}_q)$ such that $\mathcal{P} = \mathcal{F} \circ S$. While S is a randomly chosen invertible matrix, $\mathcal{F}$ must have a special structure. All m components $f_i(\mathbf{x})$ partition the variables into two sets: vinegar variables $x_0, \ldots, x_{v-1}$, which are quadratically mixed with all other variables; and oil variables $x_v, \ldots, x_{n-1}$. Visually, the matrix representations of these quadratic forms have an all-zero[3] $o \times o$ block:

$$f_i(\mathbf{x}) = \mathbf{x}^\mathsf{T} \left(\quad \right) \mathbf{x} \ . \tag{6}$$



In order to compute a signature for a document $d \in \{0,1\}^*$, the signer computes its hash $\mathbf{y} = \mathsf{H}(d)$. He then chooses a random assignment to the vinegar variables and substitutes these into the system of equations $\mathcal{P}(\mathbf{x}) = \mathbf{y}$, or more explicitly

$$\begin{cases} \vdots \\ \sum_{j=0}^{v-1} \sum_{k=0}^{j} f_{j,k}^{(i)} \underline{x_j} \underline{x_k} + \sum_{j=0}^{v-1} \sum_{k=v}^{n-1} f_{j,k}^{(i)} \underline{x_j} x_k = y_i \quad , \\ \vdots \end{cases} \tag{7}$$

where $f_{j,k}^{(i)}$ represents the coefficient of the monomial $x_j x_k$ of the ith component of $\mathcal{F}$. The underlining indicates vinegar variables, which are substituted for their assignments. It is clear from this indication that the system of equations has become linear in the remaining oil variables, and since $m = o$, it has one easily computed solution in the generic case. The signer chooses a different assignment to the vinegar variables until there is one solution. At this point, the signature $\mathbf{s} \in \mathbb{F}_q^n$ is found by computing $\mathbf{s} = S^{-1}\mathbf{x}$. It is verified through evaluation of $\mathcal{P}$, *i.e.*, $\mathcal{P}(\mathbf{s}) \stackrel{?}{=} \mathsf{H}(d)$.

4.2 Reconciliation Attack

The reconciliation attack [15] is essentially an algebraic key recovery attack: the variables are the coefficients of S^{-1} and the equations are obtained by requiring that all the polynomials be of the same form as Eqn. 6. Naïvely, this requires solving a quadratic system of $mo(o+1)$ equations in n^2 variables. However, the attack relies on the observation that there is almost always a viable S'^{-1}

[3] Or since it represents a quadratic form, skew-symmetric instead of all-zero.

compatible with (6) but of the form

$$S'^{-1} = \left(\begin{array}{c} \\ \\ \\ \\ \end{array} \right. \quad \begin{array}{c} v \\ \\ \\ \\ o \end{array} \quad \left. \begin{array}{c} \\ \\ \\ \\ \end{array} \right) \quad . \tag{8}$$

This observation is justified by the fact that only the coefficients of S^{-1} that are located in the rightmost o columns appear as indeterminates in the coefficients that are equated to zero. Moreover, any linear recombination of these columns also maps the oil-times-oil coefficients to zero and therefore we might as well consider only the representative of this equivalence class (equivalence under linear recombination of the rightmost o columns) whose bottom right $o \times o$ block is the identity matrix.

The use of this observation reduces the number of variables to $v \times o$. Moreover, the key observation behind the reconciliation attack is that the o columns of S'^{-1} can be found iteratively, solving a new quadratic system at each step. Moreover, the authors of this attack argue that the complexity of this strategy is dominated by the first step, which requires solving only m equations in v variables [15].

These optimizations are no issue in our attack on SSNE. The parameters m and n are generally small enough to make naïvely solving a quadratic system of $mo(o+1)/2$ equations in n^2 variables feasible. However, for generic systems, whenever $mo(o+1)/2 > n^2$ there might not exist a $S^{-1} \in \mathsf{GL}_n(\mathbb{F}_q)$ that brings $\mathcal{P}$ into the form of Eqn. 6. But choosing o to be different from m might bring a suitable S^{-1} back into existence. This motivates the following definition.

Definition 1 (o-reconcilable). *A system $\mathcal{P}$ of m multivariate quadratic polynomials in n variables over $\mathbb{F}_q$ is* o-reconcilable *iff there exists an $S \in \mathsf{GL}_n(\mathbb{F}_q)$ such that $\mathcal{P} \circ S$ partitions the n variables into $v = n - o$ vinegar variables and o oil variables distinguished by $\mathcal{P} \circ S$ being linear in the oil variables.*

Remark. Clearly, constant and linear terms are linear in all variables under any change of basis. Reconcilability considers only the quadratic part of the polynomials and without loss of generality we may restrict attention to their homogeneous quadratic part.

Theorem 1 (m-reconcilability of UOV). *Let $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ be the public key of a UOV cryptosystem. Then $\mathcal{P}$ is m-reconcilable.*

Proof. Trivial: follows from construction of $\mathcal{P} = \mathcal{F} \circ S$. $\mathcal{F}$ induces the required partition into oil and vinegar variables. □

Theorem 2 ($\lfloor n/2 \rfloor$-reconcilability when $m = 1$). *Assume q is odd. Let $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q$ be a single quadratic polynomial. Then $\mathcal{P}$ is $\lfloor n/2 \rfloor$-reconcilable.*

Proof. Let $Q_p \in \mathbb{F}_q^{n \times n}$ be a symmetric matrix representation of $\mathcal{P}(\mathbf{x})$ via $\mathcal{P}(\mathbf{x}) = \mathbf{x}^\mathsf{T} Q_p \mathbf{x}$. Then Q_p is diagonalizable, *i.e.*, there exists an invertible matrix $A \in \mathbb{F}_q^{n \times n}$ such that $A^\mathsf{T} Q_p A$ is nonzero only on the diagonal.

All non-zero elements on the diagonal must be one except for the last which might be the smallest quadratic non-residue in $\mathbb{F}_q$. Now choose a random symmetric matrix $Q_f \in \mathbb{F}_q^{n\times n}$ such that the lower right $\lfloor n/2 \rfloor \times \lfloor n/2 \rfloor$ block consists of all zeros and such that $\mathsf{rank}(Q_f) = \mathsf{rank}(Q_p)$. It is also diagonalizable: there is an invertible matrix $B \in \mathbb{F}_q^{n\times n}$ such that $B^{\mathsf{T}} Q_f B$ is a diagonal matrix consisting of all ones except for the last element which might be the smallest quadratic non-residue. If $B^{\mathsf{T}} Q_f B = A^{\mathsf{T}} Q_p A$ we are done because $\mathcal{F} = \mathcal{P} \circ B^{-1} \circ A$ induces the required partition. If $B^{\mathsf{T}} Q_f B \neq A^{\mathsf{T}} Q_p A$ they must differ in the last diagonal element. So then multiply any one nonzero row of Q_f by any quadratic residue and obtain another diagonalization. Now $B^{\mathsf{T}} Q_f B = A^{\mathsf{T}} Q_p A$ must hold and we are done. □

Theorem 3. *In the generic case, a system of m quadratic polynomials in n variables over $\mathbb{F}_q$ is o-reconcilable when $m(o+1)/2 \leq n$.*

Proof. The number of coefficients of S^{-1} that are involved in the $mo(o+1)/2$ equations that set the oil-times-oil coefficients to zero is no, corresponding the rightmost $n \times o$ block of S^{-1}. The other elements of S^{-1} do not affect these coefficients. This leads to a system of $mo(o+1)/2$ quadratic equations in no variables which generically has solutions when $mo(o+1)/2 \leq no$, or equivalently when $m(o+1)/2 \leq n$. □

4.3 Generating Small Solutions

After obtaining an o-reconciliation $(\mathcal{F}, S)$, the task is to obtain a solution $\mathbf{x}$ such that $\mathcal{F}(\mathbf{x}) = \mathbf{0}$ and such that $S^{-1}\mathbf{x}$ is small. The partitioning of $\mathbf{x}$ into the vinegar variables $x_0, \ldots, x_{v-1}$ and the oil variables $x_v, \ldots, x_{n-1}$ separates the shortness objective into two parts. On the one hand, the *vinegar contribution*

$$\left(S^{-1}\right)_{[:,0:(v-1)]} \mathbf{x}_{[0:(v-1)]} \tag{9}$$

must be small; on the other hand, the *oil contribution*

$$\left(S^{-1}\right)_{[:,v:(n-1)]} \mathbf{x}_{[v:(n-1)]} \tag{10}$$

must be small as well. The reason for this separation is not just that the vinegar variables and oil variables are determined in separate steps; in fact, determining vinegar variables that lead to a small vinegar contribution is easy. The form of Eqn. 8 guarantees that small vinegar variables will map onto a small vinegar contribution. Therefore, the only requirement for selecting vinegar variables is that they be small enough, say roughly $q^{1/2}$. By contrast, the process of finding suitable oil variables is far more involved.

A quadratic map where $o > m$ can be thought of as a UOV$^-$ map, *i.e.*, a UOV map with $o-m$ dropped components. This gives the signer, or an attacker who possesses the reconciliation, $o - m$ degrees of freedom for selecting the oil variables. Coupled with the freedom afforded by the choice of vinegar variables, the signer or attacker can generate a vector $\mathbf{x}$ such that $S^{-1}\mathbf{x}$ is short.

The task is thus to find an assignment to the oil variables such that a) $\mathcal{F}(\mathbf{x}) = \mathbf{0}$ is satisfied; and b) $\left(S^{-1}\right)_{[:,v:(n-1)]} \mathbf{x}_{v:(n-1)}$ is small as well. Constraint (a) is satisfiable whenever $m \leq o$ (in the generic case, *i.e.*, assuming certain square matrices over $\mathbb{F}_q$ are invertible). Constraint (b) requires $o > m$ and the resulting vector can be made shorter with growing $o - m$.

The matrix representation of a quadratic form is equivalent under addition of skew-symmetric matrices, which in particular means that it is always possible to choose an upper-triangular representation even of UOV maps such as Eqn. 6. The ith equation of $\mathcal{F}(\mathbf{x}) = \mathbf{0}$ can therefore be described as

$$f_i(\mathbf{x}) = \mathbf{x}^\mathsf{T} \left(\begin{array}{|cc} Q_i & L_i \\ \hline & \end{array} \right) \mathbf{x} + \boldsymbol{\ell}^{(i)\mathsf{T}} \mathbf{x} + c_i = 0 \tag{11}$$

$$\left(\mathbf{x}^\mathsf{T}_{[0:(v-1)]} L_i + \boldsymbol{\ell}^{(i)\mathsf{T}}_{[v:(n-1)]}\right) \mathbf{x}_{[v:(n-1)]} = -\mathbf{x}^\mathsf{T}_{[0:(v-1)]} Q_i \mathbf{x}_{[0:(v-1)]} - \boldsymbol{\ell}^{(i)\mathsf{T}}_{[0:(v-1)]} \mathbf{x}_{[0:(v-1)]} - c_i. \tag{12}$$

All m equations can jointly be described as $A\mathbf{x}_{[v:(n-1)]} = \mathbf{b}$ for some matrix $A \in \mathbb{F}_q^{m \times o}$ and vector $\mathbf{b} \in \mathbb{F}_q^m$, because the vinegar variables $\mathbf{x}_{[0:(v-1)]}$ assume constant values. Let $\mathbf{x}^{(p)}$ be any particular solution to this linear system, and let $\mathbf{x}^{(k)}_0, \ldots, \mathbf{x}^{(k)}_{o-m-1}$ be a basis for the right kernel of A. Any weighted combination of the kernel vectors plus the particular solution, is still a solution to the linear system:

$$\forall (w_0, \ldots, w_{o-m-1}) \in \mathbb{F}_q^{o-m} \,.\, A \left(\mathbf{x}^{(p)} + \sum_{i=0}^{o-m-1} w_i \mathbf{x}^{(k)}_i \right) = \mathbf{b} \ . \tag{13}$$

This means we have $o-m$ degrees of freedom with which to satisfy constraint (b).

In fact, we can use LLL for this purpose in a manner similar to the clever selection of the vinegar variables. The only difference is that the weight associated with the vector $\mathbf{x}^{(p)}$ must remain 1 because otherwise constraint (a) is not satisfied. This leads to the following application of the embedding method.

Identify $\mathbf{x}^{(p)}$ and all $\mathbf{x}^{(k)}_i$ by their image after multiplication by $\left(S^{-1}\right)_{[:,v:(n-1)]}$, thus obtaining $\mathbf{z}^{(p)} = \left(S^{-1}\right)_{[:,v:(n-1)]} \mathbf{x}^{(p)}$ and $\mathbf{z}^{(k)}_i = \left(S^{-1}\right)_{[:,v:(n-1)]} \mathbf{x}^{(k)}_i$. Then append q^2 to $\mathbf{z}^{(p)}$ and 0 to all $\mathbf{z}^{(k)}_i$, and stack all these vectors in column form over a diagonal of q's to obtain the matrix C:

$$C = \left(\begin{array}{ccc|c} — & \mathbf{z}^{(p)\mathsf{T}} & — & q^2 \\ — & \mathbf{z}^{(k)\mathsf{T}}_0 & — & 0 \\ & \vdots & & \vdots \\ — & \mathbf{z}^{(k)\mathsf{T}}_{o-m-1} & — & 0 \\ \hline q & & & \\ & \ddots & & \\ & & q & \end{array} \right) \ . \tag{14}$$

Run LLL on this matrix to obtain a reduced basis matrix $B \in \mathbb{Z}^{(o-m+1+n)\times(n+1)}$ of which the first n rows are zero, and a unimodular matrix U satisfying $B = UC$. The appended q^2 element guarantees that the row associated with the particular solution will never be added to another row because that would increase the size of the basis vectors. As a result, there will be one row in the matrix B that ends in q^2. Moreover, this row will be short because it was reduced by all other rows. We now proceed to derive an upper bound for the size of this vector considering only the first n elements, *i.e.*, without the q^2. Unfortunately, the best upper bound we can prove rigorously is $\lceil \frac{q}{2} \rceil \sqrt{n}$, but we can rely on the following heuristic argument for a meaningful result.

Let s be the index of this targeted row. Without row s and omitting the last column, the nonzero rows of B form an LLL-reduced basis for a q-ary lattice of dimension $o-m$ and embedding dimension n. We approximate the sizes of these vectors using $\lambda_i(\mathcal{L}) \approx \lambda_0(\mathcal{L})$. Coupled with the m-dimensional ball argument of Micciancio and Regev for estimating the first successive minimum [27], this gives

$$\|\mathbf{b}_\ell\|_2 \lesssim 2^{(o-m)/2} \sqrt{\frac{n}{2\pi e}} q^{(n-o+m)/n} \quad . \tag{15}$$

Moreover, row s (considered without the q^2) cannot be much larger than this quantity because it is LLL-reduced with respect to vectors of this size. So $\|\mathbf{b}_s\|_2 \approx \|\mathbf{b}_\ell\|_2$. Our experiments show that this heuristic bound is followed quite closely in practice for small m, n and large q.

The solution $\mathbf{s} = S^{-1}\mathbf{x}$ consists of two parts: the vinegar contribution and the oil contribution. Therefore, we can bound the size of the whole thing.

$$\|\mathbf{s}\|_2 \leq \|S^{-1}_{[:,0:(v-1)]}\mathbf{x}_{[0:(v-1)]}\|_2 + \|S^{-1}_{[:,v:(n-1)]}\mathbf{x}_{[v:(n-1)]}\|_2 \tag{16}$$

$$\lesssim \sqrt{n-o} \cdot q^{1/2} + 2^{(o-m)/2} \sqrt{\frac{n}{2\pi e}} q^{(n-o+m)/n} \quad . \tag{17}$$

Or if we treat n, m, o, v as small constants,

$$\|\mathbf{s}\|_2 \in O\left(q^{(n-o+m)/n}\right) \quad . \tag{18}$$

4.4 Summary

Figure 1 shows pseudocode for the algebraic-lattice hybrid attack algorithm.

Line 1 attempts to launch a UOV reconciliation attack, but the algorithm fails when this attack is unsuccessful. In fact, the criterion for success is precisely whether the map $\mathcal{P}$ is o-reconcilable. Generically, this criterion is only satisfied for $m(o+1)/2 \leq n$, as per Theorem 3, although it is certainly possible to construct maps that are o-reconcilable for $m(o+1)/2 > n$ — indeed, standard UOV public keys match this ungeneric description. A prudent strategy for maps whose structure is unknown is to try step 1 for several values of o and to pick the decomposition of $\mathcal{P}$ where o is largest. However, in this case the length of

algorithm ALHA
input: $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ — a quadratic map
: $o \in \mathbb{Z}$ — number of oil variables
output: $\mathbf{s} \in \mathbb{F}_q^n$ such that $\mathcal{P}(\mathbf{s}) = \mathbf{0}$
and such that $\|\mathbf{s}\|_2 \in O(q^{o/n} + q^{(n-o+m)/(n+1)})$

▷ find decomposition $\mathcal{P} = \mathcal{F} \circ S$ where $\mathcal{F}$ is quadratic but linear in $x_{n-o}, \ldots, x_{n-1}$, and where $S \in \mathsf{GL}_n(\mathbb{F}_q)$

1: **try:** $\mathcal{F}, S \leftarrow \mathsf{UOV\ Reconciliation\ Attack}(\mathcal{P}, o)$

▷ get vinegar variables $x_0, \ldots, x_{n-o-1}$

2: $\mathbf{x}_{[0:n-o-1]} \xleftarrow{\$} [-\lfloor q^{1/2} \rfloor : \lfloor q^{1/2} \rfloor]^{n-o}$

▷ get oil variables $x_{n-o}, \ldots, x_{n-1}$

3: Find $A \in \mathbb{F}_q^{m \times o}$ and $\mathbf{b} \in \mathbb{F}_q^m$ such that $A\mathbf{x}_{[(n-o):(n-1)]} = \mathbf{b} \Leftrightarrow \mathcal{F}(\mathbf{x}) = \mathbf{0}$
4: Find particular solution $\mathbf{x}^{(p)}$ to $A\mathbf{x}_{[(n-o):(n-1)]} = \mathbf{b}$
5: Find kernel vectors $\mathbf{x}_0^{(k)}, \ldots, \mathbf{x}_{o-m-1}^{(k)}$ of A
6: $\mathbf{z}^{(p)} \leftarrow \left(S^{-1}\right)_{[:,(n-o):(n-1)]} \mathbf{x}^{(p)}$
7: **for** $i \in [0 : (o-m-1)]$ **do:**
8: $\quad \mathbf{z}_i^{(k)} \leftarrow \left(S^{-1}\right)_{[:,(n-o):(n-1)]} \mathbf{x}_i^{(k)}$
9: **end**
10: Compile matrix C from $\mathbf{z}^{(p)}$ and $\mathbf{z}_i^{(k)}$ ▷ according to Eqn. 14
11: $U, B \leftarrow \mathsf{LLL}(C)$
12: Find s such that $B_{[s,:]}$ ends in q^2
13: $\mathbf{x}_{[(n-o):(n-1)]} \leftarrow \mathbf{x}^{(p)} + \sum_{i=0}^{o-m-1} U_{[s,1+i]} \mathbf{x}_i^{(k)}$

▷ join vinegar and oil variables, and find inverse under S

14: $\mathbf{s} \leftarrow S^{-1}\mathbf{x}$
15: **return s**

Fig. 1. Algebraic-lattice hybrid attack.

the returned solution is not fixed beforehand but depends on the largest o for which step 1 succeeds.

With this algebraic-lattice hybrid attack in mind, we formulate the last design principle for SSNE instances. The rationale is that the targeted solution should be significantly smaller (*i.e.*, κ bits, spread over n variables) than what the algebraic-lattice hybrid attack can produce.

Design Principle 6: *Let o be the largest integer such that the system is o-reconcilable. If $o > m$ then guarantee that*

$$\frac{\kappa}{n} + \log_2 \beta \leq \frac{n - o + m}{n + 1} \log_2 q \ . \tag{19}$$

4.5 Discussion

Equation 15 is an upper bound whereas we actually need a lower bound in order to delineate a portion of the parameter space where the attack does not apply. In practice, the short solutions found by the algebraic lattice hybrid attack are indeed shorter than the heuristic upper bound of Eqn. 17. Nevertheless, the solutions found by the attack have length very close to this bound, to the point where it is a suitable estimate. Fig. 2 plots in full blue the minimum length of solutions found by the algebraic lattice hybrid attack across one hundred trials for various modulus sizes. This graph follows the dashed green line, which represents the estimate or heuristic upper bound of Eqn. 17, quite closely. Both are far apart from the recommendation of design principle 6, which is drawn in full red. This graph represents many random SSNE instances with $m = 2$ and $n = 9$. The same behavior was observed across a wide range of parameter choices.

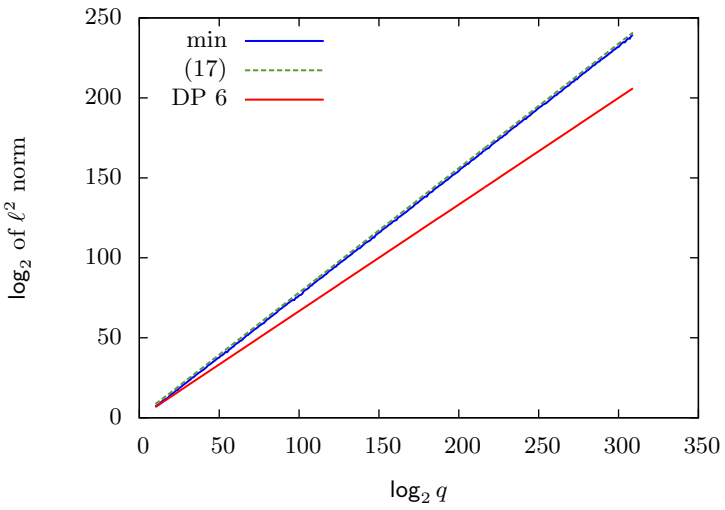


Fig. 2. Comparison of prediced length against experimental length of solutions obtained by the algebraic-lattice hybrid attack.

It is worth stressing that the algebraic-lattice hybrid attack applies only when $o > m$. When $o = m$ it does not produce solutions that are shorter than random vectors in $\mathbb{F}_q^n$, and when $o < m$ there is no guarantee it will find even one solution. Obviously, instead of requiring β to be significantly smaller than the expected length of this attack's solutions, the designer might also choose n and m so as to render the algebraic-lattice hybrid attack inapplicable.

5 Hash Function

At this time we do not know how to use SSNE to generate short-message public key functionalities. The next best option is to generate a hash function, which is what this section is about.

The resulting design does not merely exemplify using the SSNE problem constructively; it has concrete advantages over other hash functions as well. For instance, not only is the SSNE hash function provable secure (in contrast to all widely deployed hash functions), but it also relies on a *different* hard problem, which is likely to be unaffected by potential future breakthroughs in cryptanalysis of other hard problems. Also, our hash function has essentially optimal output length in terms of security: for κ bits of security against collision finders the output is 2κ bits long. This stands in contrast to many other provably secure hash functions which either have larger outputs or else require purpose-defeating post-processing functions to shrink them.

Additionally, because the hash function is built on top of SSNE instances, it requires comparably few finite field multiplications to compute. This property of having low multiplication complexity is interesting from the point of view of multiparty computation, zero-knowledge proofs, and fully homomorphic encryption, where multiplication operations are typically so expensive as to compel minimization at all costs. However, this argument ignores the cost of the bit shuffling, which are nonlinear operations over the finite field.

We note that it is possible to generate digital signature schemes from just hash functions [17,5], although the size and generation time of the signatures scales poorly. Nevertheless, anyone wanting to implement this signature scheme's key generation or signature generation procedures in a distributed manner — for instance, in order to require majority participation — must develop applied multiparty computation protocols and must consequently look to minimize multiplication complexity. Therefore, the SSNE hash function might be a good candidate for instantiating hash-based digital signature schemes with if they must enable distributed key and signature generation.

5.1 Description

We use the Merkle-Damgård construction, which requires dividing the data stream into a sequence of size b blocks. At every iteration, one data block is consumed and it is compressed with the state in order to produce a new state. The hash value is the output of the compression function after the last block has been consumed. The concept is described visually in Fig. 3.

Before applying the sequence of compression functions, the data stream $x \in \{0,1\}^*$ must first be expanded into a multiple of b bits. Let $\ell = |x|$ be the number of bits before padding, and let $\llcorner \ell \lrcorner$ be its expansion and $|\ell|$ the number of bits in this expansion. The expansion function is given by

$$\mathsf{expand} : \{0,1\}^{\ell} \to \{0,1\}^{\lceil (\ell + |\ell|)/b \rceil b} = x \mapsto x \| 0^{-\ell \bmod b} \| 0^{-|\ell| \bmod b} \| \llcorner \ell \lrcorner \ . \quad (20)$$

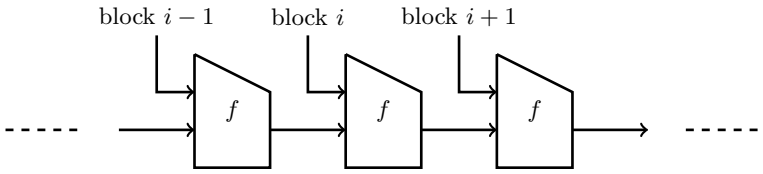


Fig. 3. Merkle-Damgård construction for hash functions.

Let q be the largest prime smaller than $2^{2\kappa}$, where κ is the targeted security level. For the purpose of defining this hash function, the elements of $\mathbb{F}_q$ are $\{0, \ldots, q-1\}$. The compression function itself decomposes into $f = \mathcal{P} \circ r$. The purpose of $r : \{0,1\}^b \times \mathbb{F}_q \to \mathbb{F}_q^2$ is to permute the bits and output two integers inside $[0 : \lceil q^{3/4} \rceil - 1]$, which are then interpreted as small elements of $\mathbb{F}_q$. In particular, on input $(s, e) \in \{0,1\}^b \times \mathbb{F}_q$, this reshuffling function takes the most significant $\frac{1}{4}\lceil \mathsf{log}_2\, q \rceil$ bits of e, appends them to s, and reinterprets this bitstring as an integer. Formally, r maps

$$r : \left(s_{b-1} \| \cdots \| s_0, \sum_{i=0}^{\lceil \mathsf{log}_2\, q \rceil - 1} 2^i e_i \right) \mapsto \left(\left(\sum_{i=0}^{b-1} 2^i s_i \right) + \left(\sum_{i=b}^{\lceil \frac{3}{4} \mathsf{log}_2\, q \rceil - 1} 2^i e_{i+b/2} \right), \sum_{i=0}^{\lceil \frac{3}{4} \mathsf{log}_2\, q \rceil - 1} 2^i e_i \right) . \tag{21}$$

In particular, this implies that $b = \frac{1}{2}\lceil \mathsf{log}_2\, q \rceil$.

The map $\mathcal{P} : \mathbb{F}_q^2 \to \mathbb{F}_q$ is a single homogeneous cubic polynomial in two variables. There are $\binom{5}{2} = 10$ coefficients which are assigned indices lexicographically from 0 to 9. Then the ith coefficient has a bit expansion equal to the first 2κ bits in the expansion of π^{i+1}, without the leading 1.

The description of the hash function is complete except for one remaining item. The initial state element, *i.e.*, the field element that is fed into the very first compression function must still be specified. For this value we choose the first 2κ bits of π^{-1}, again without the leading 1. The formal description of the algorithm is given in Fig. 4.

5.2 Security

The key property a hash function should possess is collision-resistance, which informally states that it should be difficult to find two different inputs $x, y \in \{0,1\}$ such that $\mathsf{Hash}(x) = \mathsf{Hash}(y)$. Collision-resistance implies weaker properties such as second preimage resistance and first preimage resistance (also known as one-wayness). Therefore, it suffices to show that collisions are hard to find. We demonstrate this fact by showing that any pair of colliding values implies a collision for $\mathcal{P}$, which should be difficult to find because that task requires solving a hard SSNE instance.

First, consider that expand is injective. To see this, assume there are two different strings x and y that have the same output under expand. Then $|x| \neq |y|$ because otherwise the appended tail is the same and then the difference must

algorithm Hash
input: $x \in \{0,1\}^{\ell}$ — bitstring of any length
output: $h \in \{0,1\}^{2\kappa}$ — hash value

1: $h \leftarrow \lfloor(\pi^{-1} - \frac{1}{4})2^{2\kappa+2}\rfloor$
2: $x' \leftarrow \mathsf{expand}(x)$
3: **for** $i \in [0 : |x'|/b]$ **do:**
4: $\quad e_1, e_2 \leftarrow r(x'_{[ib:(ib+b-1)]}, h)$
5: $\quad h \leftarrow \mathcal{P}(e_1, e_2)$
6: **end**
7: **return** $\llcorner h \lrcorner$

Fig. 4. Hash function relying on SSNE.

be present in their images under expand as well. However, the last b bits of the images under expand uniquely determine the length of the original strings and this quantity must be the same, which contradicts $|x| \neq |y|$. This argument assumes the length of the inputs is less than $2^b = 2^\kappa$, which is reasonable from a practical point of view. Since expand is injective, it cannot be the source of a collision.

Next, the permutation of bits r is a bijection. It cannot be the source of a collision either.

Therefore, the only source of collisions contained in the description of the hash function is $\mathcal{P}$. Finding a collision means finding a pair of vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^2$ whose elements have at most $\frac{6}{4}\kappa$ bits, such that $\mathcal{P}(\mathbf{a}) = \mathcal{P}(\mathbf{b})$. One can re-write this equation in terms of the difference $\mathbf{d}$ from the mean $\mathbf{c} = (\mathbf{a} + \mathbf{b})/2$. The equation then becomes

$$\mathcal{P}(\mathbf{c} + \mathbf{d}) - \mathcal{P}(\mathbf{c} - \mathbf{d}) = \mathbf{0} \ . \tag{22}$$

This expression is useful because its degree in $\mathbf{c}$ is one less, *i.e.*, 2 instead of 3. Therefore, by choosing a random value for $\mathbf{d}$ the attacker finds $\mathbf{c}$ by solving a *quadratic*, instead of *cubic*, SSNE instance. (In fact, this argument was precisely the motivation for a degree-3 polynomial map $\mathcal{P}$ to begin with; to kill an attack strategy that involves only finding short solutions to *linear* equations.) The parameters of the hash function were chosen to ensure that the SSNE instance of Eqn. 22 (with randomly chosen $\mathbf{d}$) satisfies all design principles.

6 Conclusion

This paper presents a new hard problem called SSNE, which is the logical merger of the SIS and MQ problems. However, in contrast to both the SIS and MQ problems, the hardness of an SSNE instance grows linearly with the size of the modulus q. This linear scaling stands in stark contrast to the quadratic and cubic

scaling of the SIS and MQ problems; and therefore, if it is possible to generate post-quantum public key cryptosystems from SSNE as it is from SIS and MQ, then these cryptosystems are very likely to require dramatically less bandwidth for having smaller public keys, ciphertexts, or signatures.

Indeed, the goal of the research that lead to the writing of this paper was to generate *public key* cryptosystems with exactly those properties. Needless to say, we have failed in that endeavor. Some of the design principles came about as a result of a process of design and attack. At least from a superficial point of view, this failure suggests that the design principles are incompatible with strategies for generating public key cryptosystems. Nevertheless, we remain hopeful about the possibility of finding strategies that are compatible with the design principles and leave their discovery as an open problem.

Acknowledgments. The authors would like to thank Fré Vercauteren and Wouter Castryck for useful discussions and references, as well as the anonymous reviewers for helpful comments. Alan Szepieniec is being supported by a Ph.D. grant from the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). This work was supported in part by the Research Council KU Leuven: C16/15/058. In addition, this work was supported by the European Commission through the Horizon 2020 research and innovation programme under grant agreement No H2020-ICT-2014-644371 WITDOM and H2020-ICT-2014-645622 PQCRYPTO.

References

1. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Miller [28], pp. 99–108, http://doi.acm.org/10.1145/237814.237838
2. Albrecht, M.R., Cid, C., Faugère, J., Fitzpatrick, R., Perret, L.: On the complexity of the BKW algorithm on LWE. Des. Codes Cryptography 74(2), 325–354 (2015), http://dx.doi.org/10.1007/s10623-013-9864-x
3. Bardet, M.: Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie. Ph.D. thesis, Pierre and Marie Curie University, Paris, France (2004), https://tel.archives-ouvertes.fr/tel-00449609
4. Bardet, M., Faugere, J.C., Salvy, B.: On the complexity of gröbner basis computation of semi-regular overdetermined algebraic equations. In: Proceedings of the International Conference on Polynomial System Solving. pp. 71–74 (2004)
5. Bernstein, D.J., Hopwood, D., Hülsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schneider, M., Schwabe, P., Wilcox-O'Hearn, Z.: SPHINCS: practical stateless hash-based signatures. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9056, pp. 368–397. Springer (2015), http://dx.doi.org/10.1007/978-3-662-46800-5_15
6. Bettale, L., Faugère, J., Perret, L.: Hybrid approach for solving multivariate systems over finite fields. J. Mathematical Cryptology 3(3), 177–197 (2009), http://dx.doi.org/10.1515/JMC.2009.009

7. Bettale, L., Faugère, J., Perret, L.: Solving polynomial systems over finite fields: improved analysis of the hybrid approach. In: van der Hoeven, J., van Hoeij, M. (eds.) International Symposium on Symbolic and Algebraic Computation, ISSAC'12, Grenoble, France - July 22 - 25, 2012. pp. 67–74. ACM (2012), http://doi.acm.org/10.1145/2442829.2442843
8. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. Lecture Notes in Computer Science, vol. 7073, pp. 1–20. Springer (2011), http://dx.doi.org/10.1007/978-3-642-25385-0_1
9. Coppersmith, D.: Finding a small root of a bivariate integer equation; factoring with high bits known. In: Maurer [25], pp. 178–189, https://doi.org/10.1007/3-540-68339-9_16
10. Coppersmith, D.: Finding a small root of a univariate modular equation. In: Maurer [25], pp. 155–165, https://doi.org/10.1007/3-540-68339-9_14
11. Coron, J.: Finding small roots of bivariate integer polynomial equations revisited. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3027, pp. 492–505. Springer (2004), https://doi.org/10.1007/978-3-540-24676-3_29
12. Coron, J.: Finding small roots of bivariate integer polynomial equations: A direct approach. In: Menezes, A. (ed.) Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4622, pp. 379–394. Springer (2007), https://doi.org/10.1007/978-3-540-74143-5_21
13. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding. Lecture Notes in Computer Science, vol. 1807, pp. 392–407. Springer (2000), http://dx.doi.org/10.1007/3-540-45539-6_27
14. Ding, J., Yang, B.Y.: Multivariate public key cryptography. In: Post-quantum cryptography, pp. 193–241. Springer (2009)
15. Ding, J., Yang, B., Chen, C.O., Chen, M., Cheng, C.: New differential-algebraic attacks and reparametrization of rainbow. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) Applied Cryptography and Network Security, 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5037, pp. 242–257 (2008), https://doi.org/10.1007/978-3-540-68914-0_15
16. Faugere, J.C.: A new efficient algorithm for computing gröbner bases (F 4). Journal of pure and applied algebra 139(1), 61–88 (1999)
17. Goldreich, O.: The Foundations of Cryptography - Volume 2, Basic Applications. Cambridge University Press (2004)
18. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Miller [28], pp. 212–219, http://doi.acm.org/10.1145/237814.237866
19. Howgrave-Graham, N.: Finding small roots of univariate modular equations revisited. In: Darnell, M. (ed.) Cryptography and Coding, 6th IMA International Conference, Cirencester, UK, December 17-19, 1997, Proceedings. Lecture Notes

in Computer Science, vol. 1355, pp. 131–142. Springer (1997), https://doi.org/10.1007/BFb0024458
20. Jao, D., Feo, L.D.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B. (ed.) Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings. Lecture Notes in Computer Science, vol. 7071, pp. 19–34. Springer (2011), http://dx.doi.org/10.1007/978-3-642-25405-5_2
21. Jutla, C.S.: On finding small solutions of modular multivariate polynomial equations. In: Nyberg, K. (ed.) Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding. Lecture Notes in Computer Science, vol. 1403, pp. 158–170. Springer (1998), https://doi.org/10.1007/BFb0054124
22. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: Stern, J. (ed.) Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding. Lecture Notes in Computer Science, vol. 1592, pp. 206–222. Springer (1999), http://dx.doi.org/10.1007/3-540-48910-X_15
23. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261(4), 515–534 (Dec 1982), https://doi.org/10.1007/BF01457454
24. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261(4), 515–534 (Dec 1982), https://doi.org/10.1007/BF01457454
25. Maurer, U.M. (ed.): Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding, Lecture Notes in Computer Science, vol. 1070. Springer (1996), https://doi.org/10.1007/3-540-68339-9
26. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM J. Comput. 37(1), 267–302 (2007), http://dx.doi.org/10.1137/S0097539705447360
27. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 147–191. Springer Berlin Heidelberg, Berlin, Heidelberg (2009), https://doi.org/10.1007/978-3-540-88702-7_5
28. Miller, G.L. (ed.): Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996. ACM (1996)
29. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005. pp. 84–93. ACM (2005), http://doi.acm.org/10.1145/1060590.1060603
30. Ritzenhofen, M.: On efficiently calculating small solutions of systems of polynomial equations: lattice-based methods and applications to cryptography. Ph.D. thesis, Ruhr University Bochum (2010), http://www-brs.ub.ruhr-uni-bochum.de/netahtml/HSS/Diss/RitzenhofenMaike/
31. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994. pp. 124–134. IEEE Computer Society (1994), http://dx.doi.org/10.1109/SFCS.1994.365700