Cryptanalysis of the Square Cryptosystems

Olivier Billet and Gilles Macario-Rat

Orange Labs, Issy-les-Moulineaux, France
billet@eurecom.fr, gilles.macariorat@orange-ftgroup.com

Abstract. Following the cryptanalyses of the encryption scheme HFE and of the signature scheme SFLASH, no serious alternative multivariate cryptosystems remained, except maybe the signature schemes UOV and HFE^{--}. Recently, two proposals have been made to build highly efficient multivariate cryptosystems around a quadratic *internal* transformation: the first one is a signature scheme called square-vinegar and the second one is an encryption scheme called square introduced at CT-RSA 2009. In this paper, we present a total break of both the square-vinegar signature scheme and the square encryption scheme. For the practical parameters proposed by the authors of these cryptosystems, the complexity of our attacks is about 2^{35} operations. All the steps of the attack have been implemented in the Magma computer algebra system and allowed to experimentally assess the results presented in this paper.

1 Introduction

There are mainly two motivations behind the construction of multivariate cryptosystems. The original one is to provide alternatives to the asymmetric schemes RSA and those based on Discrete Logarithm problems which are connected to number theoretic problems. Multivariate cryptosystems are instead connected to the hardness of solving randomly chosen systems of multivariate equations over a finite field, a problem which is NP-complete even in the case of quadratic polynomials defined over GF(2) when there are at least two such polynomials in the system. Moreover, this problem seems to be hard not only for very special instances but also on the average. Another incentive to develop multivariate cryptosystems is the expected efficiency that they might offer, a property that would be highly appreciated for constrained environments such as RFIDs and other embedded devices. Finally, some people argue about the fact that, contrary to the problem of factorisation and that of solving discrete logarithms [23], no quantum algorithm is known for the problem of solving sets of randomly chosen multivariate equations.

After the introduction of the C^* cryptosystem by Matsumoto and Imai in [13, 16], there have been several other proposals. Among the most famous ones are certainly HFE (Hidden Field Equations) and SFLASH which can be thought of as two ways of generalising the C^* scheme. Some heuristic design principles have followed. A major one, which has been originally suggested by Shamir in [21], is to remove some equations from the public mapping in the case of signature

schemes; this principle has proven to be successful in thwarting Patarin's attack [17] against C^* (an attack that can be viewed as a preliminary to Gröbner basis attacks). Another one consists in adding a new set of variables to perturb the analysis as in the UOV (Unbalanced Oil and Vinegar) signature scheme [14].

Two of the most promising proposals, SFLASH and HFE have been cryptanalysed during the last years. Some HFE instances have been shown to succumb to Gröbner basis attacks in [7] and the complexity of such attack has been argued to be quasi-polynomial in [12]. SFLASH has been entirely broken: the missing equations (due to the minus transformation) can be recovered in most cases as explained in [6] and the secret key of the resulting C^* scheme can be recovered following the cryptanalysis described in [10]. In this context, two new proposals were based on internal transformations that are not only quadratic on the base field, but also on the extension field: a signature scheme called square-vinegar was proposed in [2] and an encryption scheme called square appears in [4].

Our Results. In this paper, we expose a total break of both the square-vinegar signature and the square encryption proposals from a theoretical point of view as well as from a practical point of view. We indeed describe how to recover an equivalent secret key for both cryptosystems given the public key alone. For the parameters recommended by the authors, our attacks complete in a few minutes on a standard PC. These cryptanalyses also represent a theoretical break of the schemes as, under some reasonable assumptions, their complexity is shown to be polynomial with respect to the security parameter: the attacks have a time complexity of $O\big(\log^2(q)n^6\big)$ since they rely on standard linear algebra on n^2 unknowns over a finite field of size q and n is typically small because the time complexity of the public computation (signature or encryption) is $O(n^3)$. The attacks are sequences of steps including the discovery of new algebraic invariants leaking from the public key, a careful analysis of these invariants to sort out vinegar unknowns from the standard ones. We additionally implemented Magma [3] programs that were used to verify each of the steps of the cryptanalyses and to perform the attacks against the different sets of parameters recommended by the designers of the square encryption and square-vinegar signature schemes. Their source code is given in the appendix.

2 The Square Cryptosystems

The square cryptosystems are based on design ideas taken from both the HFE cryptosystem and the UOV cryptosystem. However, an important property of the square cryptosystems is that they are defined over fields of *odd* characteristic: as their internal transformations are quadratic, the systems would be linear over fields of characteristic 2. We begin by a brief reminder on HFE and UOV before proceeding to the description of the square cryptosystems themselves.

2.1 The HFE Cryptosystem

The HFE cryptosystem has been proposed by Patarin in [18] as a possible generalisation (and strengthening) of the C^* scheme proposed by Matsumoto and

Imai in [16]. Indeed C^* was broken by Patarin [17], whereas the best attack against HFE are Gröbner basis attacks which complexity was argued to be quasi-polynomial [7, 12]. HFE is called hidden field equation because its internal transformation is kept secret. This internal transformation F is defined over an extension $\mathbb{E}$ of degree n over some base field $\mathbb{F}_q$ and is chosen to be $\mathbb{F}_q$-quadratic:

$$F:\ X \mapsto \sum_{\substack{0\leq i<j<n \\ q^i+q^j\leq D}} \alpha_{i,j} X^{q^i+q^j} + \sum_{\substack{0\leq k<n \\ q^k\leq D}} \beta_k X^{q^k} + \gamma\ , \tag{1}$$

where the coefficients $\alpha_{i,j}$, β_k, and γ lie in $\mathbb{E}$ and D is an upper bound to the overall degree to make it practical to invert F through factorization. Since F is a $\mathbb{F}_q$-quadratic mapping, it can also be expressed over $\mathbb{F}_q$ as an n-tuple $(f_1, \ldots, f_n)$ of quadratic polynomial mappings in n unknowns and so can the composition $T \circ F \circ S$ for any pair of one-to-one affine mappings $S : \mathbb{F}_q^n \to \mathbb{E}$ and $T : \mathbb{E} \to \mathbb{F}_q^n$. In the case of HFE, the mappings S and T are kept secret and together with F, constitute the secret key, whereas the public key is the mapping $G = T \circ F \circ S$. In order to decrypt, the legitimate user applies the inverse of T, finds roots of the univariate polynomials on the extension field $\mathbb{E}$ and applies the inverse of S to each of these roots. The plaintext is one of the roots which can be singled out by using some redundancy. In this decryption process, the knowledge of the secrets S and T is crucial.

Additionally, Shamir's proposal to remove some (say r) of the n polynomials that constitutes the public key can be applied in the case of a signature scheme: indeed, to sign a message $(y_1, \ldots, y_{n-r})$, the signer first completes the message with random values y_{n-r+1}, ..., y_n and "decrypts" it normally. This operation is called the minus transformation and is used in the square-vinegar scheme.

With these notations, C^* is similar to HFE (with an unbounded total degree) where all coefficients of the internal transformation are set to zero but $\alpha_{0,\theta}$ for a well chosen θ. SFLASH in turn [1], is the original C^* scheme with the minus transformation applied.

2.2 The UOV Signature Scheme

Another ingredient in the design of the square-vinegar signature scheme is the use of additional unknowns meant to harden the analysis of the scheme by trying to break the structure used during the decryption process. Such an idea was first proposed in the oil and vinegar signature scheme. This scheme uses two sets of unknowns $(x_1, \ldots, x_n)$ and $(z_1, \ldots, z_v)$ respectively called the oil and the vinegar variables. The internal transformation then consists of an n-tuple of polynomials $F = (f_1, \ldots, f_n)$ of the special form:

$$f_i(x,z) = \sum_{\substack{1\leq i\leq n \\ 1\leq j\leq v}} \alpha_{i,j} x_i z_j + \sum_{1\leq i\leq n} \beta_i x_i + \sum_{1\leq i\leq v} \gamma_i z_i + \sum_{1\leq i\leq j\leq v} \delta_{i,j} z_i z_j + \epsilon\ , \tag{2}$$

where $\alpha_{i,j}$, β_i, γ_i, $\delta_{i,j}$, and ϵ are randomly chosen from the base field $\mathbb{F}_q$. The x_i are called oil variables because they do not mix, i.e. there is no cross-term $x_i x_j$.

Vinegar variables z_i in contrast, mix with other vinegar variables as well as with oil variables. The fact that the coefficients of the polynomials are chosen randomly is satisfactory since the resulting polynomials look closer to randomly chosen ones. However, the two types of variables makes it possible to create a signature scheme: in order to find some pre-image $y = (y_1, \ldots, y_n)$ through F the signer first draws some random values for $z_1, \ldots, z_v$ and substitutes them in the description of F. The resulting set of polynomials becomes linear in the oil unknowns x_i and the associated $n \times n$ linear system (with y as right member) is easily solved: about $\frac{1}{e}$ of the time, the system has a solution $(a_1, \ldots, a_n)$ which makes (a, z) a pre-image of y through F and otherwise another choice for z is made until there is a solution. Obviously, this structure has to be hidden from the view of an attacker and the public key is the composition $G = F \circ S$ where $S : \mathbb{F}_q^{n+v} \to \mathbb{F}_q^{n+v}$ is a one-to-one affine application.

The message size over signature size for the UOV signature scheme is not optimal since the number of vinegar unknowns must be at least twice big as the number of oil unknowns for it to be secure [22, 19, 14].

2.3 The Square-Vinegar Signature Scheme

The square-vinegar signature scheme strives to provide an efficient alternative to UOV or HFE with the minus transformation applied. Let $\mathbb{F}_q$ be a finite field and $\mathbb{E}$ be an extension of degree n over $\mathbb{F}_q$. The internal transformation of the square-vinegar scheme is defined as:

$$F \ : \ \mathbb{E} \times \mathbb{F}_q^v \ \longrightarrow \ \mathbb{E} \ , \quad (X, X_v) \ \longmapsto \ \alpha X^2 + \beta(X_v)X + \gamma(X_v) \ , \tag{3}$$

where α is a constant randomly chosen from $\mathbb{E}$, $\beta : \mathbb{F}_q^v \to \mathbb{E}$ is a randomly chosen affine application, and $\gamma : \mathbb{F}_q^v \to \mathbb{E}$ is a randomly chosen $\mathbb{F}_q$-quadratic application. This internal transformation is hidden by two full rank affine applications $S : K^{n+v} \to \mathbb{E}$ and $T : \mathbb{E} \to \mathbb{F}_q^n$. Therefore S mixes the vinegar unknowns X_v with the "normal" unknowns X. In addition to T, a projection Π is applied where r of the n components have been removed as in SFLASH or HFE^{--}. The affine transforms S and T together with the applications γ, β, and the constant α constitute the secret key. The public key P results from the composition of the three applications: $P = \Pi \circ T \circ F \circ S$.

The use of an odd characteristic base field is advertised by the authors as a means to thwart Gröbner bases attacks since introducing the corresponding field equations in the computation renders it unpractical. Mixing the vinegar unknowns with the normal ones breaks the algebraic relations between the input and the output that appeared in C^* (bilinear relations [17]) or HFE (algebraic relations of higher degree, as explained in [7, 12]). Eventually, just as for HFE^{--}, removing part of the output information further mitigates Gröbner bases attacks and prevents Kipnis and Shamir's attack developed against UOV.

Signature. The signing process is highly efficient. It only requires the holder of the secret key to randomly pick r elements from $\mathbb{F}_q$ to complete the message $(m_1, \ldots, m_{n-r})$ to be signed into $\tilde{m} = (m_1, \ldots, m_{n-r}, \tilde{m}_{n-r+1}, \ldots, \tilde{m}_n)$

and to invert the public application in three steps: $S^{-1} \circ F^{-1} \circ T^{-1}(\tilde{m})$. Applying T^{-1} and S^{-1} is a matter of multiplying with precomputed matrices and inverting F requires to find the roots of a quadratic univariate polynomial over $\mathbb{E}$. In case there is no solution, the signer restarts the process by choosing another way of completing the message m into $\tilde{m}$.

2.4 The Square Encryption Scheme

A companion scheme to this square-vinegar signature scheme has been proposed in [4]. The square encryption scheme strives to provide an efficient and secure alternative to HFE and, as the square-vinegar scheme, has a square internal transformation: $F : \mathbb{E} \to \mathbb{E}$, $X \mapsto X^2$. The parameters are chosen so that the size of the base field verifies $q \equiv 3 \mod 4$ and the degree n of $\mathbb{E}$ over $\mathbb{F}_q$ is odd. The transformation F is again hidden by two full rank affine mappings $S : \mathbb{F}_q^{n-r} \to \mathbb{E}$ and $T : \mathbb{E} \to \mathbb{F}_q^n$, which yields a public key $P = T \circ F \circ S$. (Following [5], the authors proposed to fix r of the input unknowns to a predefined value (say, zero) to prevent the attacker from controlling the differential of the public key as in Dubois, Fouque, Shamir, and Stern's cryptanalysis [6].) This scheme is somewhat reminiscent of the C^* scheme, where $F(X) = X^{q^\theta+1}$ for a well chosen θ. But for the square encryption where $\theta = 0$, the bilinear relations $XY^{q^\theta} = X^{q^{2\theta}}Y$ between X and $Y = F(X)$ boils down to the tautology $XY = YX$. The embedding S aims to finish hiding the algebraic structure of the internal transformation.

Decryption. The secrets' holder is able to decrypt very efficiently: in addition to finding pre-images through T and S which amounts to solve simple linear systems, the decryption process requires to compute a square root in the extension field $\mathbb{E}$. Computing the square root is done by the square and multiply algorithm $X = Y^{\frac{q^n+1}{4}}$ since $q^n \equiv 3 \mod 4$. As there are two possible square roots, the right one is singled out as the one lying in the image of S.

3 Cryptanalysis of the Square-Vinegar Signature Scheme

We now describe a generic and very efficient attack against the square-vinegar signature scheme. Our attack proceeds in three steps: We first exhibit an invariant of the internal transformation and recover it through the analysis of the differential of the public key; Then, we use this information to recover an equivalent representation of the vinegar space; In a third step, we transform the public key into a special shape that allows us to invert it efficiently. Put together, these three steps allow us to forge a signature for any given message.

3.1 Alternative Decompositions

Recall that the internal transformation of the square-vinegar signature scheme has the following structure:

$$F : \mathbb{E} \times \mathbb{F}_q^v \longrightarrow \mathbb{E} \ , \quad (X, X_v) \longmapsto \alpha X^2 + \beta(X_v)X + \gamma(X_v) \ ,$$

where α is a constant, $\beta : \mathbb{F}_q^v \to \mathbb{E}$ is an affine $\mathbb{F}_q$-linear mapping, and $\gamma : \mathbb{F}_q^v \to \mathbb{E}$ is a $\mathbb{F}_q$-quadratic mapping, where $\mathbb{E}$ is an extension of degree n over $\mathbb{F}_q$. The public key is the mapping $P = \Pi \circ T \circ F \circ S$, where Π is a projection that removes r polynomials, $S : \mathbb{F}_q^{n+v} \to \mathbb{E} \times \mathbb{F}_q^v$ and $T : \mathbb{E} \to \mathbb{F}_q^{n-r}$ are two affine linear mappings of full rank. The decomposition (T, F, S) of the public key is kept secret.

A major component of the internal transformation F is the mixing of vinegar unknowns with X. It makes it harder for an attacker to use the specific structure of a univariate quadratic polynomial of F viewed as a function of X. A crucial remark is that there exist linear mappings that, when composed with the internal transformation, not only conserve its special form, but also discard the part of F mixing the vinegar X_v with X. Indeed, consider the mappings $\sigma : (X, X_v) \mapsto \left(X - \frac{\alpha}{2}\beta(X_v), X_v\right)$ and $\tau : Y \mapsto \frac{1}{\alpha}Y$. (Remember that the scheme is defined over a field $\mathbb{F}_q$ of odd characteristic.) It can be checked that these mappings provide an alternative decomposition $(T \circ \tau, \tilde{F}, \sigma \circ S)$ of the public key such that

$$\tilde{F} : (X, X_v) \mapsto X^2 + \tilde{\gamma}(X_v) \ , \tag{4}$$

where $\tilde{\gamma}$ is a $\mathbb{F}_q$-quadratic mapping. We stress here that an attacker does not need to know the mappings σ and τ but rather assumes without loss of generality that the public key follows the specific decomposition (4). (Also note that in a similar fashion, keeping secret the defining polynomial of the extension has no effect: as two fields of the same size are isomorphic and the isomorphism is a linear bijective application, any arbitrary choice made by the attacker is "absorbed" in S and T.) This last decomposition can be further tweaked as in [11] to remove the affine parts of the mappings S and T but at the expense of reintroducing a linear term in X, leading to an internal transformation of the following shape:

$$F' : (X, X_v) \mapsto X^2 + \beta' X + \gamma'(X_v) \ , \tag{5}$$

where β' is a *constant* from $\mathbb{E}$ and γ' is some $\mathbb{F}_q$-quadratic mapping. In the following sections, the attacker can therefore just assume wlog that the public key is decomposed as (T', F', S') where S' and T' are linear mappings, and F' is as given in (5): then (T', F', S') contains enough information to forge valid signatures and thus constitutes an equivalent secret key. We call such a decomposition a "split decomposition" (the unknowns X and X_v are now separated in the internal transformation). A split decomposition is not unique: iterates of the Frobenius mapping $\varphi : z \mapsto z^q$ and multiplications $\Lambda_u : z \mapsto uz$, $u \in \mathbb{E}$, do not alter the prescribed shape of the internal transformation (though coefficients might change); In particular, if (T_0, F_0, S_0) is a split decomposition, so are $(T_0 \circ \Lambda_{u^{-2}}, \Lambda_{u^2} \circ F_0 \circ \Lambda_{u^{-1}}, \Lambda_u \circ S_0)$ and $(T_0 \circ \varphi^{-i}, \varphi^i \circ F_0 \circ \varphi^{-i}, \varphi^i \circ S_0)$.

3.2 Using the Multiplicative Property of the Differential

In the previous section we showed how to discard the cross-contribution of X_v and X. However, the contribution $\gamma(X_v)$ still disturbs the algebraic properties

of the univariate quadratic in X. In order to circumvent this difficulty, we make use of a tool first introduced by Fouque *et al.* in [9] that proved very useful in attacking multivariate cryptosystems: the differential of the public mapping. The differential of P in a is defined as: $\mathrm{D}P_a(x) = P(x+a) - P(x) - P(a) + P(0)$.

In the case of an $\mathbb{F}_q$-quadratic mapping, $\mathrm{D}P_a$ is such that $(x, a) \mapsto \mathrm{D}P_a(x)$ is a symmetric bilinear mapping. From now on, we denote by $\mathrm{D}P$ this bilinear mapping and call it differential of P. The differential map corresponding to the internal transformation $X \mapsto X^2 + \beta X + \gamma(X_v)$ of a square-vinegar instance is:

$$\mathrm{D}F\big((X, X_v), (Y, Y_v)\big) = 2XY + \mathrm{D}\gamma(X_v, Y_v) \ . \tag{6}$$

The success of the attack lies in the fact that normal (X) and vinegar (X_v) unknowns are separated in the expression of the differential $\mathrm{D}F$. More precisely, the only linear mappings L such that for all (X, X_v) and all (Y, Y_v):

$$\mathrm{D}F\big((L(X), X_v), (Y, Y_v)\big) - \mathrm{D}F\big((X, X_v), (L(Y), Y_v)\big) = 0 \ \Leftrightarrow \ L(X)Y = YL(X)$$

are $Z \mapsto \lambda Z$ for $\lambda \in \mathbb{E}$. Indeed, any solution $L : Z \mapsto \sum_{1 \leq i < n} l_i Z^{q^i}$ verifies $\sum_{1 \leq i < n} l_i XY^{q^i} = \sum_{1 \leq i < n} l_i X^{q^i} Y$ for all X and Y, and since $(X, Y) \mapsto X^{q^i} Y^{q^j}$ forms a basis of the space of bilinear forms we must have $l_i = 0$ for all $i > 0$.

In addition, we conjecture that with very high probability (with respect to the uniform choice of the coefficients of γ) the only linear mappings L verifying

$$\forall X_v \, \forall Y_v \quad \mathrm{D}\gamma\big(L(X_v), Y_v\big) - \mathrm{D}\gamma\big(X_v, L(Y_v)\big) = 0$$

are $Z_v \mapsto cZ_v$ for some $c \in \mathbb{F}_q$. This might be heuristically justified by the fact that the random choice of γ does not allow such an algebraic property to appear, and is verified experimentally. Assuming this conjecture is true, we have:

Proposition 1. *For a random instance of the square-vinegar scheme, it happens with very high probability that the only linear mappings L verifying:*

$$\forall (X, X_v) \forall (Y, Y_v) \quad DF\big(L(X, X_v), (Y, Y_v)\big) - DF\big((X, X_v), L(Y, Y_v)\big) = 0 \tag{7}$$

are $(Z, Z_v) \mapsto (\lambda Z, cZ_v)$, where $\lambda \in \mathbb{E}$ and $c \in \mathbb{F}_q$.

Proof. Write $L : (Z, Z_v) \mapsto (AZ + CZ_v, \tilde{C}Z + BZ_v)$ for some solution of (7). Since the equation holds for all inputs of $\mathrm{D}F$, consider it specialised at $X_v = 0$ and $Y_v = 0$, with $\mathrm{D}F$ replaced by its expression (6):

$$\forall X \, \forall Y \quad \big[2A(X)Y + \mathrm{D}\gamma(\tilde{C}(X), 0)\big] - \big[2XA(Y) - \mathrm{D}\gamma(0, \tilde{C}(Y))\big] = 0 \ .$$

As $\mathrm{D}\gamma(*, 0) = 0$ and $\mathrm{D}\gamma(0, *) = 0$ for any '$*$', this gives $A(X)Y = XA(Y)$ which, as we saw above, implies $A : Z \mapsto \lambda Z$ for $\lambda \in \mathbb{E}$. Similarly, at $X = 0$ and $Y = 0$, (7) becomes: $\forall X_v \forall Y_x \ \ \mathrm{D}\gamma\big(B(X_v), Y_v\big) - \mathrm{D}\gamma\big(X_v, B(Y_v)\big) = 0$, implying $B : Z \mapsto cZ$ for $c \in \mathbb{F}_q$ by conjecture. Finally, at $X = 0$ and $Y_v = 0$, (7) becomes: $\forall X_v \forall Y \ \ \mathrm{D}\gamma\big(X_v, \tilde{C}(Y)\big) = 2C(X_v)Y$. Assume for a contradiction that C is not identically null. Then setting $X_v = x_1$ such that $C(x_1) \neq 0$, the right hand side

spans a vector space of dimension n while the left hand side spans a vector space of dimension at most v. Hence, when $v < n$ as in a square-vinegar instance, C must be identically null. Then, for all (X_v, Y), we have $\mathrm{D}\gamma(X_v, \tilde{C}(Y)) = 0$ or equivalently $\gamma(X_v + \tilde{C}(Y)) = \gamma(X_v) + \gamma(\tilde{C}(Y))$. In particular, this holds for $X_v = \tilde{C}(X)$ for any X and any Y so that $Z \mapsto \gamma(\tilde{C}(Z))$ is affine, that is, γ is affine over $\mathrm{Im}(\tilde{C})$. For a random γ it is improbable that γ is affine over some (non-zero) sub-space. Hence, with high probability, $\tilde{C}$ is identically null. □

This property of F naturally transports to the public key, provided the removal of polynomials do not completely destroy its algebraic structure:

Claim 1. *If the number of coordinates removed by the projection Π is less than half and the coefficients of γ are randomly chosen, the set of linear mappings L satisfying*

$$\forall X \, \forall Y \quad \mathrm{D}P\big(L(X), Y\big) - \mathrm{D}P\big(X, L(Y)\big) = 0$$

is $\{S^{-1} \circ \Lambda_{u,c} \circ S\}_{u \in \mathbb{E}, c \in \mathbb{F}_q}$, i.e. the conjugates by the secret mapping S of all the multiplications $\Lambda_{u,c} : (X, X_v) \mapsto (uX, cX_v)$, where $u \in \mathbb{E}$ and $c \in \mathbb{F}_q$.

3.3 Extracting the Vinegar Vector Space

The solution set Σ of Claim 1 can be easily determined as it amounts to solve a linear system of $(n-r)(n+v)^2$ equations in the $(n+v)^2$ unknowns of L over a finite field of size q. Let us call "vinegar vector space" the image through S of all the values v such that the n first coordinates of $S(v)$ are zero. Similarly, let us call "normal vector space" the image through S of all the values v such that the v last coordinates equal zero. Before explaining how to use the knowledge of Σ to recover these two vector spaces, let us state three useful lemmas.

Lemma 1. *Let u be in $\mathbb{E}$, π_u be the minimal polynomial of u over $\mathbb{F}_q$, and $\chi_{\Lambda_{u,c}}$ be the characteristic polynomial of $\Lambda_{u,c} : (X, X_v) \mapsto (uX, cX_v)$. Then:*

$$\chi_{\Lambda_{u,c}}(x) = (x-c)^v \cdot \pi_u(x)^{\frac{n}{\deg \pi_u}} \ .$$

Lemma 2. *Let u be in $\mathbb{E}$ and π_u the minimal polynomial of u over $\mathbb{F}_q$. Then:*

$$\pi_u(x) = (x-u)(x-u^q)\cdots\left(x-u^{q^{\deg(\pi_u)-1}}\right) \ .$$

Lemma 3 (Thm. 3.25 [15]). *The number of irreducible monic polynomials of degree n in $\mathbb{F}_q[X]$ is $\frac{1}{n}\sum_{d|n} \mu(d) q^{\frac{n}{d}}$ where μ is the Möbius function.*[1] *It follows that the number of elements in $\mathbb{E}$ with a minimal polynomial of degree n is at least $q^n - q^{\frac{n}{2}} - q^{\frac{n}{2}-1} - \cdots - q^2 - q$.*

Let M be any element picked at random from the solution set Σ of Claim 1. Since $M = S^{-1} \circ \Lambda_{u,c} \circ S$ for some $(u, c) \in \mathbb{E} \times \mathbb{F}_q$, M and $\Lambda_{u,c}$ are conjugate and thus have the same characteristic polynomial $\chi_M(x) = (x-c)^v \cdot \pi_u(x)^{\frac{n}{\deg \pi_u}}$

[1] $\mu(1) = 1$, $\mu(x) = (-1)^k$ for x a product of k distinct primes, and $\mu(x) = 0$ otherwise

according to Lemma 1. In addition, Lemma 3 shows that for u chosen uniformly at random in $\mathbb{E}$, $\deg(\pi_u)$ has more than $1-q/q^{\frac{n}{2}}$ chances to be n. We can therefore assume in the following that c and π_u are known from the factorization of χ.

The factorization of π_u over $\mathbb{E}$ in turn discloses u^{q^i} for some unknown i. However, as stated at the end of Section 3.1, the split decomposition is not affected by iterates of the Frobenius mapping and thus it is enough to solve for $\tilde{S}$ in the following linear system:

$$\tilde{S} \circ M = \Lambda_{u^{q^i},c} \circ \tilde{S} \ .$$

Any particular solution S_0 of this system is sufficient, since the whole space of solutions is a coset of the commutant of $\Lambda_{u^{q^i},c}$. The commutant of $X \mapsto u^{q^i}X$ is the space of multiplications, since u does not belong to any subfield of $\mathbb{E}$. On the contrary, the commutant of $X_v \mapsto cX_v$ is the whole space of $\mathbb{F}_q$-linear mappings, since precisely c lies in $\mathbb{F}_q$. At this point, the attacker is almost in the same position as the legitimate signer to produce a signature since he has access to the vinegar space through S_0 and can now work on

$$P \circ S_0^{-1}(X, X_v) = \Pi \circ T \circ \left(X^2 + \beta X + \gamma(X_v)\right)$$

instead of the original public key P. Let us define $\tilde{P} = P \circ S_0^{-1}$.

The next step of the attack is to recover a mapping equivalent to T. To this end, we seek to cancel the part of $\tilde{P}$ that is linear in X which can be achieved by using an adequate change of variables $X \mapsto (X-b)$, where b is to be determined. The expression of $\tilde{P}(X-b)$ with respect to X in turn contains a quadratic part, a linear part, and a constant part. Looking at the linear part alone, the attacker writes down that the set a coefficients of X are equal to zero; these coefficients are a set of $(n-r)$ affine functions with respect to b and solving for b allows the attacker to recover β. The final step is to recover an equivalent version of T. This is done by considering the part of $\tilde{P}$ that is quadratic with respect to X: $Q(X) = \Pi \circ T(X^2)$. By composing with multiplications over $\mathbb{E}$, it is possible to complete the $(n-r)$ coordinates of Q into a full set $\tilde{Q}(X)$ of n coordinates by taking a basis of $\{Q(\lambda X)\}_{\lambda \in \mathbb{E}}$. Then, solving for $\tilde{T}$ in $\tilde{Q}(X) = \tilde{T}(X^2)$ gives an equivalent representation T_0 of T.

At this point, the attacker gained the knowledge of S_0, T_0, and β_0 such that:

$$P \circ S_0^{-1}(X, X_v) = \Pi \circ T_0 \circ (X^2 + \beta_0 X) + P \circ S_0^{-1}(0, X_v) \ .$$

We claim that this is equivalent to the knowledge of the secret key since the attacker is then able to sign any message m as efficiently as the legitimate signer as follows. Draw some random value X_v from the vinegar space and randomly complete the $(n-r)$ coordinates of $m - P \circ S_0^{-1}(0, X_v)$ into an n coordinates value $\tilde{m}$. Compute $Y = T_0^{-1}(\tilde{m})$ and solve for X_0 in $(X + \frac{1}{2}\beta_0)^2 = Y + \frac{1}{4}\beta_0^2$. A signature of m is then given by $S_0^{-1}(X_0, X_v)$.

3.4 Complexity Analysis and Practical Parameters

Our attack requires $O\big(\log^2(q)(n+v)^6\big)$ operations to find the solution set Σ of Claim 1 and $O\big(\log^2(q)(n+v)^3\big)$ operations to factor the characteristic polyno-

mial χ. The particular solution S_0 is found with $O\big(\log^2(q)(n+v)^6\big)$ operations. The complexity of the other steps can be neglected and thus the attack has an overall complexity of $O\big(\log^2(q)(n+v)^6\big)$.

The authors of the square-vinegar signature scheme claimed a 80-bits security for the following parameter sets:

	parameter set 1	parameter set 2
field size q	31	13
normal unknowns n	31	36
vinegar unknowns v	4	4
removed polynomials r	3	3

The complexity of our attack is about 2^{35} and our Magma program in appendix completes within minutes for both parameter sets on a common desktop PC.

4 Cryptanalysis of the Square Encryption Scheme

The square encryption scheme poses new challenges to the attacker. Its design strategy of embedding the plaintext into a bigger space before applying the internal transformation makes it impossible to use the differential mapping as was done previously. This is due to the restricted view the attacker has on the input space which does not allow to manipulate the inner of the differential easily. In our attack against the square encryption scheme, we therefore use a different technique. Instead of peeling off the cryptosystem from the input, we peel it off from the output.

4.1 Equivalent Representation of the Secret Key

Due to the specific form of the internal transformation and without loss of generality, we may give the following alternative decomposition of the public key:

$$P(X) = T\big(S(X)^2\big) + T\big(s \cdot S(X)\big) + t \ , \tag{8}$$

where S and T are the linear part of the original secret linear mappings and $s = \frac{1}{2}\sigma$ and $t = \tau + T(\sigma^2)$ with σ and τ the original secret constants from $\mathbb{E}$. Since the mappings S and T are linear, it can be easily seen that with respect to the input X, the first term of (8) is $\mathbb{F}_q$-quadratic, the second term is linear, and the third term is constant. Furthermore, these three homogeneous terms can be read directly on the public key itself, so that the attacker knows the following:

$$P_2(X) = T\big(S(X)^2\big) \ , \qquad P_1(X) = T\big(s \cdot S(X)\big) \ , \qquad P_0(X) = t \ .$$

4.2 Looking for Invariant Subspaces

As with the signature scheme, the differential of the public key provides useful information to the attacker. In the case of the square encryption scheme, it can be expressed as:

$$\mathrm{D}P(X,Y) = T\big(2 \cdot S(X) \cdot S(Y)\big) \ .$$

Let consider the partial mappings $\mathrm{D}P_y : X \mapsto \mathrm{D}P(X, y)$. Since $S : \mathbb{F}_q^{n-r} \to \mathbb{E}$ has full rank, its image is of dimension $(n-r)$. Hence, choosing any linearly independent vectors $y_1, \ldots, y_{n-r}$ makes $\mathrm{D}P_{y_1}, \ldots, \mathrm{D}P_{y_{n-r}}$ span the whole vector space of mappings $\{\mathrm{D}P_z\}_{z\in\mathbb{E}}$. This shows that the attacker is able to derive a set of mappings $\Delta = \{P_1\} \cup \{\mathrm{D}P_{y_i}\}_{i=1,\ldots,n-r}$ each of which has the special form $T \circ \Lambda_\alpha \circ S$, where Λ_α stands for the multiplication by α in $\mathbb{E}$. This set of mappings can then be rewritten as $\Delta = \{T \circ \Lambda_{\lambda_i} \circ S\}_{i=1,\ldots,n-r+1}$ where the $n-r+1$ values $\lambda_1, \ldots, \lambda_{n-r+1}$ are unknown, but linearly independent.

The attacker does not need to know the actual value of the λ_i since he can exploit this set of mappings in as follows. The general idea is to look for linear mappings L that can link the public equations, say two elements $D_1 = T \circ \Lambda_{\lambda_1} \circ S$ and $D_2 = T \circ \Lambda_{\lambda_2} \circ S$ from Δ. One natural idea is then to look for L such that:

$$L \circ D_1 = D_2 \ , \tag{9}$$

since it can be easily checked that $L_0 = T \circ \Lambda_{\lambda_2\lambda_1^{-1}} \circ T^{-1}$ is a particular solution of (9). However, the solution space of (9) is not restricted to multiplications. This is due to the 'embedding' mechanism, i.e. the fact that the mapping S is not a one-to-one mapping, which release some of the constraints and allows less structured linear mapping to be solutions.

A possible direction to solve this issue is to put more constraints on the mapping L while being careful to keep mappings of the form $T \circ \Lambda_* \circ T^{-1}$ in the solution space. This is why we not only look for a linear mapping that solves (9), but *several* equations similar to (9) simultaneously. This can be reformulated in terms of Δ as follows. We look for linear mappings L such that:

$$\forall i \in \{1, \ldots, m\}, \qquad L \circ (T \circ \Lambda_{\lambda_i} \circ S) \in \left\langle T \circ \Lambda_{\lambda_{m+1}} \circ S, \ldots, T \circ \Lambda_{\lambda_{n-r+1}} \circ S \right\rangle \ , \tag{10}$$

that is, the image through L of m elements of Δ must lie in the vector space spanned by the remaining elements of Δ. It is easy to see that if λ is such that:

$$\forall i \in \{1, \ldots, m\}, \qquad \lambda \cdot \lambda_i \in \left\langle \lambda_{m+1}, \ldots, \lambda_{n-r+1} \right\rangle \ , \tag{11}$$

then $T \circ \Lambda_\lambda \circ T^{-1}$ must be solution of (10).

The parameter m controls the number of solutions of (10) and (11). It can be used to simultaneously render system (11) under-determined and system (10) over-determined. This ensures that no other solutions except than the conjugates of multiplications. We can determine suitable values of m as follows. For $i \leq m$, the fact that $\lambda \cdot \lambda_i$ lies in $\langle \lambda_{m+1}, \ldots, \lambda_{n-r+1} \rangle$ puts $n - ((n-r+1) - m)$ constraints on the n coordinates of λ in $\mathbb{F}_q$. As $\lambda_1, \ldots, \lambda_{n-r+1}$ are linearly independent, the above constraints are independent. Hence (11) admits solutions as soon as $n > m(n - (n-r+1-m))$. Similarly, the whole space of linear mappings L has dimension n^2 and each equation of (10) puts $n(n-r) - (n-r+1-m)$ constraints as mappings from Δ map $\mathbb{F}_q^{n-r}$ to $\mathbb{F}_q^n$. Therefore, system (10) is over-determined as soon as $n^2 \leq m\big(n(n-r) - (n-r+1-m)\big)$. These two conditions define a range of values of m such that the solution space of (10) becomes isomorphic to the solution space of (11). This behavior is entirely confirmed by our Magma implementation of the attack.

4.3 Recovery of the Secret Elements

Once a linear mapping $L = T \circ \Lambda_\lambda \circ T^{-1}$ has been recovered, every element of the secret key can be computed. By proceeding just as for the signature scheme, the underlying multiplication λ is revealed from the characteristic polynomial of L. An equivalent representation T_0 of T is then recovered by solving for $\tilde{T}$ in $\tilde{T} \circ L = \Lambda_\lambda \circ \tilde{T}$. Let a be a randomly chosen element. The other component of the secret key can then be found via:

$$S(a) = \sqrt{T_0^{-1}\big(P_2(a)\big)} \ , \qquad s_0 = \frac{1}{S(a)} \cdot T_0^{-1}\big(P_1(a)\big) \ , \qquad S_0 = \frac{1}{s_0} \cdot T_0^{-1} \circ P_1 \ .$$

(In the case where $T_0^{-1}\big(P_2(a)\big)$ is not a square in $\mathbb{E}$, just replace T_0 by $-T_0$.)

4.4 Practical Parameters

The most time consuming step of our attack is to compute the solution space of (10) which requires $O\big(\log^2(q)n^6\big)$ operations. The authors of the square encryption scheme claimed a 80-bit security for the following parameter sets:

	parameter set 1	parameter set 2
field size q	31	31
unknowns $n - r$	34	51
polynomials n	37	54

but the complexity of our attack actually is about 2^{36} operations for the first parameter set and about 2^{39} for the second. Again, the key recovery written in Magma only requires a couple of seconds to complete on a standard workstation. During the attack, $m = 2$ was enough in practice to ensure that only conjugates of multiplications were solutions.

References

1. M.-L. Akkar, N. Courtois, L. Goubin, and R. Duteuil. A Fast and Secure Implementation of Sflash. In Y. G. Desmedt, ed., *Public Key Cryptography—PKC 2003*, vol. 2567 of *LNCS*, pp. 267–278. Springer-Verlag, 2003.
2. J. Baena, C. Clough, and J. Ding. Square-vinegar signature scheme. In J. Buchmann and J. Ding, ed., *Post-Quantum Cryptography—PQCrypto 2008*, vol. 5299 of *LNCS*, pp. 17–30. Springer, 2008.
3. W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997.
4. C. Clough, J. Baena, J. Ding, B.-Y. Yang, and M.-S. Chen. Square, a New Multivariate Encryption Scheme. In M. Fischlin, ed., *Topics in Cryptology—CT-RSA 2009*, vol. 5473 of *LNCS*, pp. 252–264. Springer, 2009.
5. J. Ding, C. Wolf, and B.-Y. Yang. ℓ-Invertible Cycles for Multivariate Quadratic Public Key Cryptography. In T. Okamoto and X. Wang, eds., *Public Key Cryptography—PKC 2007*, vol. 4450 of *LNCS*, pp. 266–281. Springer, 2007.

6. V. Dubois, P.-A. Fouque, A. Shamir, and J. Stern. Practical Cryptanalysis of SFLASH. In A. Menezes, ed., *Advances in Cryptology—CRYPTO 2007*, vol. 4622 of *LNCS*, pp. 1–12. Springer, 2007.
7. J.-C. Faugère and A. Joux. Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems using Gröbner Bases. In D. Boneh, ed., *Advances in Cryptology—CRYPTO 2003*, vol. 2729 of *LNCS*, pp. 44–60. Springer, 2003.
8. J.-C. Faugère and L. Perret. Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. In S. Vaudenay, ed., *Advances in Cryptology—EUROCRYPT 2006*, vol. 4004 of *LNCS*, pp. 30–47. Springer, 2006.
9. P.-A. Fouque, L. Granboulan, and J. Stern. Differential cryptanalysis for multivariate schemes. In R. Cramer, ed., *Advances in Cryptology—EUROCRYPT 2005*, vol. 3494 of *LNCS*, pp. 341–353. Springer, 2005.
10. P.-A. Fouque, G. Macario-Rat, and J. Stern. Key Recovery on Hidden Monomial Multivariate Schemes. In N. P. Smart, ed., *Advances in Cryptology—EUROCRYPT 2008*, vol. 4965 of *LNCS*, pp. 19–30. Springer, 2008.
11. W. Geiselmann, R. Steinwandt, and T. Beth. Attacking the Affine Parts of SFLASH. In B. Honary, editor, *Cryptography and Coding—IMA 2001*, volume 2260 of *Lecture Notes in Computer Science*, pages 355–359, 2001.
12. L. Granboulan, A. Joux, and J. Stern. Inverting HFE is Quasipolynomial. In C. Dwork, ed., *Advances in Cryptology—CRYPTO 2006*, vol. 4117 of *LNCS*, pp. 345–356. Springer, 2006.
13. H. Imai and T. Matsumoto. Algebraic Methods for Constructing Asymmetric Cryptosystems. In J. Calmet, ed., *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes—AAECC 3*, vol. 229 of *LNCS*, pp. 108–119. Springer, 1985.
14. A. Kipnis, J. Patarin, and L. Goubin. Unbalanced Oil and Vinegar Signature Schemes. In J. Stern, ed., *Advances in Cryptology—EUROCRYPT '99*, vol. 1592 of *LNCS*, pp. 206–222. Springer, 1999.
15. R. Lidl and H. Niederreiter. *Finite fields*, vol. 20 of *Encyclopedia of mathematics and its applications*. Cambridge university press, 2003.
16. T. Matsumoto and H. Imai. Public Quadratic Polynominal Tuples for Efficient Signature Verification and Message Encryption. In C. G. Günther, ed., *Advances in Cryptology—EUROCRYPT '88*, vol. 330 of *LNCS*, pp. 419–453. Springer, 1988.
17. J. Patarin. Cryptoanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt '88. In D. Coppersmith, ed., *Advances in Cryptology—CRYPTO '95*, vol. 963 of *LNCS*, pp. 248–261. Springer, 1995.
18. J. Patarin. Hidden fields equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In U. M. Maurer, ed., *Advances in Cryptology—EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 33–48. Springer, 1996.
19. J. Patarin. The Oil and Vinegar Algorithm for Signatures. Presented at the Dagsthul Workshop on Cryptography, September 1997.
20. J. Patarin, L. Goubin, and N. T. Courtois. Improved Algorithms for Isomorphisms of Polynomials. In K. Nyberg, ed., *Advances in Cryptology—EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 184–200. Springer, 1998.
21. A. Shamir. Efficient Signature Schemes Based on Birational Permutations. In D. R. Stinson, ed., *Advances in Cryptology—CRYPTO '93*, vol. 773 of *LNCS*, pp. 1–12. Springer, 1993.
22. A. Shamir and A. Kipnis. Cryptanalysis of the Oil & Vinegar Signature Scheme. In H. Krawczyk, ed., *Advances in Cryptology—CRYPTO '98*, vol. 1462 of *LNCS*, pp. 257–266. Springer, 1998.
23. P. W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J.Sci.Stat.Comp.*, 26:1484, 1997.

A Simple Auxiliary Functions for our Magma Scripts

Simple functions. The following function returns a root of $ax^2 + bx + c$.

```
1  SOLVE_2ND_DEGREE:=function(a,b,c)
2    is_,sqrt_delta:=ISSQUARE(b^2-4*a*c);
3    return is_,(is_ select (-b+sqrt_delta)/(2*a) else 0);
4  end function;
```

Juggling between matrices and vectors:

```
5  MAT2VEC:=func< MAT | VECTOR(ELTSEQ(MAT)) >;
6  VEC2MAT:=func< vect, ncol | MATRIX(ncol, ELTSEQ(vect)) >;
```

SPACE returns the vector space spanned by a set of matrices MS viewed as vectors:

```
7  SPACE:=func< MS, KK, dim |
8      sub<VECTORSPACE(KK, dim)|[MAT2VEC(MS[i]) : i in [1..#MS]]> >;
```

The following returns the matrix of $x \mapsto \lambda x$:

```
9  MULBY:=func< λ, ETOV, VTOE, B |
10                 MATRIX([ETOV(VTOE(B[i])*λ) : i in [1..#B]]) >;
```

Sequences of coefficients. It can be convenient to represent a quadratic polynomial as sequences of coefficients of its homogeneous degree 0, 1, and 2 components. C_{012} takes a function P viewed as a sequence of *n_pol* polynomials on *n_var* variables and outputs the corresponding sequences CS0, CS1, and CS2:

```
11 C012:=function(KK, V_INPUT, P, n_pol, n_var)
12   CS0:=[KK ! 0:ii in [1..n_pol]];
13   CS1:=[[KK ! 0:i in [1..n_var]]:ii in [1..n_pol]];
14   CS2:=[[[KK ! 0:j in [1..i]]:i in [1..n_var]]:ii in [1..n_pol]];
15   x:=V_INPUT ! 0; y:=P(x);
16   for ii:=1 to n_pol do CS0[ii]:=y[ii]; end for;    // constant
17   for i:=1 to n_var do
18     x:=V_INPUT ! 0; x[i]:=KK ! 1; y1:=P(x); x[i]:=KK ! -1; y2:=P(x);
19     for ii:=1 to n_pol do
20       CS1[ii][i]:=(y1[ii]-y2[ii])*(KK ! 2)^-1;    // coefficient of x_i,
21       CS2[ii][i][i]:=(y1[ii]+y2[ii])*(KK ! 2)^-1-CS0[ii];    // and x_i^2,
22     end for;
23   end for;
24   for i:=2 to n_var do for j:=1 to i-1 do
25       x:=V_INPUT ! 0; x[i]:=KK ! 1; x[j]:=KK ! 1; y:=P(x);
26       for ii:=1 to n_pol do
27         CS2[ii][i][j] := y[ii]-CS2[ii][i][i]-CS2[ii][j][j]-
28               CS1[ii][i]-CS1[ii][j]-CS0[ii];    // and x_i x_j, i ≠ j
29       end for;
30   end for; end for;
31   return CS0, CS1, CS2;
32 end function;
```

Given three sequences of coefficients C_0, C_1, and C_2 defined with respect to a quadratic polynomial P as above, compute the value taken by P on input x:

```
33  Eval:=func< C2, C1, C0, x, n_var |
34          &+[ &+[C2[i][j]*x[i]*x[j] : j in [1..i]] : i in [1..n_var]]
35                  + &+[C1[i]*x[i]:i in [1..n_var]] + C0 >;
```

The next function computes the coefficients of the differential associated to the homogeneous form of degree 2 specified by the sequence of its coefficients:

```
36  Diff:=function(CS2, KK, n_pol, n_var)
37    DP:=[ZeroMatrix(KK, n_var, n_var): ii in [1..n_pol]];
38    for ii:=1 to n_pol do
39      for i:=1 to n_var do
40        DP[ii][i,i]:=2*CS2[ii][i][i];
41        for j:=1 to i-1 do
42          DP[ii][i,j]:=CS2[ii][i][j]; DP[ii][j,i]:=CS2[ii][i][j];
43        end for; end for; end for;
44  return DP; end function;
```

B Magma Script to Attack the Signature Scheme

An extension E of degree n over the base field K, also viewed as vector space V:

```
45  q:=31; n:=31; v:=4; r:=3; K:=GF(q); E:=ext<K|n>;
46  V,E2V:=VectorSpace(E,K); V2E:=E2V^-1;
```

```
47  V_input:=VectorSpace(K,n+v); V_vinegar:=VectorSpace(K,v);
48  V_message:=VectorSpace(K,n-r); V_random:=VectorSpace(K,r);
```

We then randomly draw a secret key: the coefficient α, the linear mapping β, and the quadratic mapping γ to form the internal transformation

$$F : (X, X_v) \mapsto \alpha X^2 + \beta(X_v)X + \gamma(X_v) \ ,$$

```
49  α:=V2E(A[1]) where A is Random(GL(n,K));    // ensures α ≠ 0
```

```
50  β0:=Random(E); β1:=[Random(E):i in [1..v]];
51  β:=func< Xv | &+[β1[i]*Xv[i]:i in [1..v]] + β0 >;
```

```
52  γ0:=Random(E); γ1:=[Random(E):i in [1..v]];
53  γ2:=[[Random(E):j in [1..i]]:i in [1..v]];
54  γ:=func< Xv |
55     &+[ &+[γ2[i][j]*Xv[i]*Xv[j] : j in [1..i]] : i in [1..v]] +
56              &+[γ1[i]*Xv[i]:i in [1..v]] + γ0 >;
```

```
57  F:=func< X,Xv | α*X^2+β(Xv)*X+γ(Xv) >;
```

and randomly draw input and ouput linear layers S and T:

```
58  S1:=Random(GL(n+v,K)); S0:=Random(V_input);
59  T1:=Random(GL(n,K)); T0:=Random(V);
```

The corresponding public key is obtained via $P = T \circ F \circ S$:

```
60  P:=function(input)
61    XX:=input*S1+S0;
62    X:=V2E(VECTOR([XX[i]:i in [1..n]]));    // normal variables
63    Xv:=VECTOR([XX[i]:i in [n+1..n+v]]);    // vinegar variables
64    return E2V( F(X,Xv) )*T1+T0;
65  end function;
```

The coefficients of homogeneous parts for the set of forms corresponding to P is obtained via:

```
66  PUBC0, PUBC1, PUBC2:=C012(K, V_INPUT, P, n-r, n+v);
```

We are now able to verify if a signature is valid:

```
67  VERIFY:=function(msg, sig)
68    m:=[ EVAL(PUBC2[i], PUBC1[i], PUBC0[i], sig, n+v) : i in [1..n-r]];
69    return &and[ m[i] eq msg[i]: i in [1..n-r]];
70  end function;
```

We now compute an equivalent secret key. First, we look for the linear mappings Mx verifying: $\mathsf{Mx} \times DP - DP \times \mathsf{Mx} = 0$.

```
71  B:=BASIS(V); PR:=POLYNOMIALRING(K,(n+v)^2);
72  MX:=MATRIX(n+v, [PR.i:i in [1..(n+v)^2]]);
73  DP:=DIFF(PUBC2,K,n-r,n+v);
74  EQS:=[ELTSEQ(MX*DP[ii]-DP[ii]*TRANSPOSE(MX)):ii in [1..n-r]];
75  GB:=[];
76  for ii:=1 to n-r do
77    GB:=GROEBNERBASIS(GB cat EQS[ii]);
78    if #GB + n + 1 eq (n+v)^2 then break; end if;
79  end for;
```

We choose a particular solution M_ by removing the $n+1$ degrees of freedom by fixing the remaining unknowns to random values, and extract the two roots $c \in K$ and $a \in E$ of the characteristic polynomial of M_.

```
80  repeat W:=GROEBNERBASIS([PR.((n+v)^2-i) + RANDOM(K):i in [0..n]] cat GB);
81  until not(W eq [PR ! 1]);    // complete consitently
82  M_:=MATRIX(n+v,[K ! EVALUATE(W[i],PR.i,0):i in [1..(n+v)^2]]);
83  CPOL:=FACTOREDCHARACTERISTICPOLYNOMIAL(M_);
84  if not(#CPOL eq 2) then "Bad Char. Pol."; exit; end if;
85  c:=ROOTS(CPOL[1][1])[1][1];    // factor of degree 1
86  a:=ROOTS(POLYNOMIALRING(E) ! CPOL[2][1])[1][1];    // of degree n
```

M_ must be similar to the matrix of $(X, X_v) \mapsto (aX, cX_v)$, which will disclose a particular solution S_ as useful to sign as S:

```
87  A:=MULBY(a, E2V, V2E, B);
88  is_similar,S_:=ISSIMILAR(M_, DIAGONALJOIN(A, SCALARMATRIX(v,c)) );
89  if not(is_similar) then "Recovering S_ failed."; exit; end if;
```

Applying the change of base S_, we get $(Z, Z_v) \mapsto T\big(Z^2 + \tilde{\beta} \cdot Z + \tilde{\gamma}(Z_v)\big)$:

```
90  Z:=Vector([PR.i:i in [1..n+v]])*MatrixAlgebra(PR, n+v) ! (S_);
91  PubZ:=[Eval(PubC2[i], PubC1[i], PubC0[i], Z, n+v):i in [1..n-r]];
```

To get rid of the term $\tilde{\beta} \cdot Z$, we look for Y such that the coefficient of Z in $T\big((X+Y)^2 + \tilde{\beta} \cdot (Z+Y) + \tilde{\gamma}(Z_v)$ becomes zero:

```
92  V0:=Random(V_vinegar);
93  ZpY:=[PR ! 0:i in [1..(n+v)^2]];    // Z + Y
94  for i:=1 to n do ZpY[i]:=PR.i+PR.(i+n+v); end for;
95  for i:=1 to v do ZpY[i+n]:=V0[i]; end for;
96  PubZv:=[Evaluate(PubZ[i], ZpY):i in [1..n-r]];
97  OY:=[PR ! 0:i in [1..(n+v)^2]];    // (Z, Y) = (0, Y)
98  for i:=1 to n do OY[i+n+v]:=PR.(i+n+v); end for;
99  EqLin:=&cat[[Evaluate(Coefficient(PubZv[i], PR.j, 1), OY)
100                 :j in [1..n]]:i in [1..n-r]];    // equations 2Y = β̃
101 Y0:=GroebnerBasis(EqLin);
102 beta_:=Vector([K ! Evaluate(Y0[i], PR.(i+n+v), 0):i in [1..n]]);
```

We are now able to get the polynomials corresponding to $T\big(Z^2 + \bar{\gamma}(Z_v)\big)$:

```
103 for i:=1 to n do ZpY[i]:=PR.i-beta_[i]; end for;
104 PubZ0:=[Evaluate(PubZ[i], ZpY):i in [1..n-r]];
```

We recover $g_0 = \bar{\gamma}(0)$ (remember vinegar part of ZpY was set to zero above):

```
105 g0:=[K ! Evaluate(PubZ0[i], [PR ! 0:i in [1..(n+v)^2]]):i in [1..n-r]];
```

and thus $Z \mapsto T\big(Z^2\big)$ together with its differential $(X, Y) \mapsto 2XY$

```
106 PubZ2:=[PubZ0[i]-g0[i]:i in [1..n-r]];
107 DPubZ2:=[Submatrix(S_*DP[i]*Transpose(S_), 1, 1, n, n):i in [1..n-r]];
```

but also $(X, Y) \mapsto 2a^2XY$:

```
108 DPubZa:=[A*DPubZ2[i]*Transpose(A):i in [1..n-r]];
```

This allows us to complete T into a full rank mapping T_ via $\mathsf{T\_}(X) = \frac{1}{2}DP(X, 1)$:

```
109 SPa:=Space(DPubZ2 cat DPubZa, K, n*n); SP2:=Space(DPubZ2, K, n*n);
110 W:=Basis(Complement(SPa, SP2));
111 DPplus:=DPubZ2 cat [Vec2Mat(W[i], n) : i in [1..#W]];
112 T_:=(K ! 2)^-1*Matrix([Vector([(B[i]*DPplus[j], B[1])
113                                 :j in [1..n]]) : i in [1..n]]);
```

and to forge a signature for any message:

```
114 msg:=Random(V_message);
115 repeat
116 Y:=Vector(Eltseq(msg-Vector(g0)) cat Eltseq(Random(V_random)));
117 is_square, sqrX:=IsSquare( V2E(Y*T_^-1) ); until is_square;
118 forged:=Vector(Eltseq( E2V(sqrX)-beta_ ) cat Eltseq(V0))*S_;
119 if Verify(msg, forged) then "Forged signature."; end if;
```

C Magma Script to Attack the Encryption Scheme

```
120  q:=31; n:=37; r:=3; K:=GF(q); E:=ext<K|n>;
121  VI:=VectorSpace(K, n-r); VO, K2V:=VectorSpace(E, K); V2K:=K2V^-1;
```

Build the secret key, the encryption function P, and coefficients:

```
122  L1:=Submatrix(Random(GL(n, K)), 1, 1, n-r, n); L2:=Random(GL(n, K));
123  l1:=Random(GL(n, K))[1]; l2:=Random(VO);
124  Pencrypt:=func< plain | K2V(V2K(plain*L1+l1)^2)*L2+l2 >;
125  PubC0, PubC1, PubC2 := C012(K, VI, Pencrypt, n, n-r);
```

The mappings $\Delta = \{\mathrm{D}P_{y_i}\}_{i\in[1,n-r]}$ for linearly independant $y_1, \ldots, y_{n-r}$:

```
126  DP:=Diff(PubC2, K, n, n-r); Y:=Random(GL(n-r, K));
127  Δ:=[Transpose(Matrix([Y[k]*DP[i] : i in [1..n]])) : k in [1..n-r]];
```

The set Λ of linear mappings verifying (9) for some parameter m:

```
128  m:=2; δ:=[Δ[i]: i in [m+1..n-r]]; SP:=Space(δ, K, (n-r)*n);
129  Dual:=Transpose(NullspaceMatrix(Transpose(BasisMatrix(SP))));
130  P1:=Transpose(Matrix(PubC1)); B:=Basis(VectorSpace(K, n^2));
131  MMul:=func<A | Matrix([Mat2Vec(A*Vec2Mat(B[i], n)): i in [1..#B]])>;
132  Λ:=&meet[Nullspace(MMul(Δ[i])*Dual): i in [1..m]]
133        meet Nullspace(MMul(P1)*Dual);
```

Compute the characteristic polynomial CP of a random linear mapping in Λ:

```
134  M:=Vec2Mat(Random(Λ), n); CP:=FactoredCharacteristicPolynomial(M);
135  a:=Roots(PolynomialRing(E) ! CP[1][1])[1][1];
136  A:=MulBy(a, K2V, V2K, Basis(VO));
```

Recover the secret elements:

```
137  res, L2_:= IsSimilar(M, A); R:=Random(VI);
138  v:=V2K(Vector([(R*DP[j], R): j in [1..n]])*L2_^-1)/2;
139  res, s:=IsSquare(v);
140  if not res then L2_:=-L2_; res, s:=IsSquare(-v); end if;
141  l1_:=K2V(V2K(R*P1*L2_^-1)/(2*s));
142  L1_:=P1*L2_^-1*MulBy(1/V2K(2*l1_), K2V, V2K, Basis(VO));
143  l2_:=Pencrypt(VI ! 0)-K2V(V2K(l1_)^2)*L2_;
144  IML1_:=sub<VO|[L1_[i]:i in [1..n-r]]>;

145  Disclose:=function(cipher)    // unlegitimate decryption!
146    is_square, root:=IsSquare(V2K((cipher-l2_)*L2_^-1));
147    if is_square then Z:=K2V(root);
148          if (Z-l1_) in IML1_ then return true, Solution(L1_, Z-l1_);
149      else if (-Z-l1_) in IML1_ then return true, Solution(L1_, -Z-l1_);
150      else return false, _; end if; end if; else return false, _; end if;
151  end function;
152  plain:=Random(VI); b, p:=Disclose(Pencrypt(plain));
153  if b and (p eq plain) then "Decryption successful."; end if;
```