Popping "R-propping": breaking hardness assumptions for matrix groups over $\mathbb{F}_{2^8}$

Fernando Virdia*

Abstract. A recent series of works (Hecht, IACR ePrint, 2020–2021) propose to build post-quantum public-key encapsulation, digital signatures, group key agreement and oblivious transfer from "R-propped" variants of the Symmetrical Decomposition and Discrete Logarithm problems for matrix groups over $\mathbb{F}_{2^8}$. We break all four proposals by presenting a linearisation attack on the Symmetrical Decomposition platform, a forgery attack on the signature scheme, and a demonstration of the insecurity of the instances of the Discrete Logarithm Problem used for signatures, group key agreement and oblivious transfer, showing that none of the schemes provides adequate security.

1 Introduction

The potential advent of quantum computers has kicked off the search for computational problems that resist cryptanalysis via quantum algorithms. While most of the focus on design and cryptanalysis has been put into a small set of hardness assumption families related to lattice-based, isogeny-based, and code-based computational problems [BBD09], alternative less popular proposals have been made.

One such example is a recent line of work [Hec20a,Hec20b,Hec21a,Hec21b,Hec21c] that proposes to build various post-quantum primitives based on computational problems over the ring of $k \times k$ matrices over $\mathbb{F}_{2^8}$, $M_k(\mathbb{F}_{2^8})$.[1] In particular, the authors suggest building ElGamal signatures [ElG85], Burmester-Desmedt group key agreement[2] [BD94] and Chou-Orlandi 1-out-n oblivious transfer [CO15] out of the Discrete Logarithm Problem (DLP) over $M_k(\mathbb{F}_{2^8})$. They also propose building a public-key encryption (PKE) scheme similar to the HK17 [HK17] proposal submitted to the standardisation process for post-quantum cryptography run by the US National Institute of Standards and Technology (NIST), basing it on the hardness of the Generalised Symmetrical Decomposition Problem over $M_k(\mathbb{F}_{2^8})$.

In this paper, we show that none of the schemes achieves meaningful security by describing and implementing general attacks against the GSDP and DLP instances suggested, and by describing a general forgery attack on the signature scheme that is independent of the hardness of the DLP. We remark that the forgery attack is a result of a flaw in the scheme presented in [Hec21a] which is not present in the original ElGamal signature scheme [ElG85].

Roadmap. In Section 2 we give general preliminaries and notation. In Section 3 we describe the GSDP hardness assumption and the proposed PKE schemes from [Hec20a,Hec20b], and present a plaintext-recovery linearisation-based attack. In Section 4 we describe a forgery attack on the proposed signature scheme from [Hec21a] that is independent of the hardness of the DLP. In Section 5 we use a classical result from the literature on cryptanalysis of the DLP over matrices to show that it does not provide post-quantum security and that the proposed instances of the DLP from [Hec21a,Hec21b,Hec21c] do not provide any classical security either.

2 Preliminaries

Notation. We denote by $\mathbb{Z}$ the ring of integers $\{\dots, -2, -1, 0, 1, 2, \dots\}$, by $[n]$ the set $\{1, \dots, n\}$, and by $\mathbb{Z}_q$ the ring of integers "modulo q", $\mathbb{Z}/q\mathbb{Z}$. We denote by $\mathbb{F}_{q^n}$ the finite field of q^n elements,

*The work of Virdia was carried out while employed by ETH Zürich.

[1] The authors of [Hec20a,Hec20b,Hec21a,Hec21b,Hec21c] refer to this approach as "R-propping", where "R-" stands for Rijndael, suggesting that the security of the schemes is partly achieved by using the same field used to describe the AES S-box.

[2] Called "conference key distribution" in [BD94].

where q is prime. If $n = 1$, then $\mathbb{F}_q = \mathbb{Z}_q$. We note that $\mathbb{F}_{q^n}$ can be constructed as $\mathbb{Z}_q[x]/\langle p(x)\rangle$, where $\mathbb{Z}_q[x]$ is the ring of polynomials in x with coefficients in $\mathbb{Z}_q$, p is a monic irreducible polynomial of degree n and $\langle p(x)\rangle \subset \mathbb{Z}_q[x]$ is the ideal generated by p. Given a ring R, we define the ring $(R^{r\times c}, +, \times)$ of matrices with r rows and c columns, and entries over R, with the usual commutative matrix addition and non-commutative matrix multiplication operations $+$ and $\times$. Whenever $r = c$, we may write $M_n(R)$ to denote the $R^{r\times c}$ ring. When multiplying matrices A and B, we will omit $\times$ and write AB instead. Given a matrix A, we denote its entry on the i^{th} row and j^{th} column by $(A)_{i,j}$. Given a square matrix A, we denote by $\det(A)$ its determinant. Given a polynomial p, we denote by $\deg(p)$ its degree. Given a finite set S, we write $|S|$ to mean the cardinality of S, and we denote by $x \leftarrow\!\!\$\ S$ an element x sampled uniformly at random from S. Given bit strings s_1, s_2 of the same length, we denote by $s_1 \oplus s_2$ the bit-wise exclusive-or of the strings.

Algebra. In the remainder of this work we will work in particular with the finite field of 256 elements. One way of constructing $\mathbb{F}_{2^8}$ is by using the polynomial $p(x) = 1 + x + x^3 + x^4 + x^8$, the same famously used to define the AES S-box [AES01].

Definition 1 (Discrete Logarithm Problem). *Let G be a group. Given $(x, y) \in G \times G$, such that $y = x^\ell$ for some $\ell \in \mathbb{Z}$, find $\ell \bmod ord(x)$, where $ord(x)$ is the order of the subgroup generated by x.*

3 Plaintext-recovery attack

The Generalised Symmetrical Decomposition Problem (GSDP) is a family of computational problems introduced in [CDW07], proposed as hardness assumptions that could be used to build key exchange and public-key encryption (PKE) from non-commutative groups.

Definition 2 (Generalized Symmetrical Decomposition Problem [CDW07]). *Let G be a non-commutative group. Given $(x, y) \in G \times G$, $S \subset G$ and $m, n \in \mathbb{Z}$, find $z \in S$ such that $y = z^m\, x\, z^n$.*

Remark 1. We note that the GSDP problem can also be similarly defined over rings with non-commutative multiplication.

In [Hec20a,Hec20b], the authors propose two PKE schemes that make use of the GSDP assumption over matrix groups. Both of these are extensions of HK17 [HK17]. The new schemes propose using matrices over $\mathbb{F}_{2^8}$ to avoid possible linearisation attacks [Hec20b].

We show a linearisation attack on [Hec20a,Hec20b] that allows a passive eavesdropper to decrypt any ciphertext by having access to the ciphertext itself and the public key.

Remark 2. We note that the attacks in this section result in plaintext-recovery without directly solving the GSDP instances generated by the scheme. This indicates a gap between the security of the PKE scheme and the (conjectured) hardness of the GSDP instances.

3.1 HK17-like Public-Key Encryption

We proceed to define the PKE scheme proposed in [Hec20b]. The scheme is almost identical to the one in [Hec20a] (see Definition 3), and both [Hec20a,Hec20b] are equally affected by our plaintext recovery attack. All the schemes in [HK17,Hec20a,Hec20b] are themselves based on the scheme proposed in [CDW07, § 4.4].

Definition 3. *Let u, k, d be positive integers. Let $pp = (u, k, d)$ be a tuple of public parameters. Let $\mathcal{M} = \{0,1\}^\ell$ be the message space for some finite ℓ, and let $H : \mathbb{F}_{2^8}^{k\times k} \to \mathcal{M}$ be a random oracle. In Figure 1 we define three algorithms:*

- *A probabilistic algorithm Gen(pp) that takes public parameters pp and outputs a public and private key pair (pk, sk).*
- *A probabilistic algorithm Enc(pk, m) that takes a public key pk and a message m and outputs a ciphertext c.*

Gen(pp)	Enc(pk, m)
$A \leftarrow\!\!\$\ \mathbb{F}_{2^8}^{k\times k}$	$(A, B, r) \leftarrow pk$
$B \leftarrow\!\!\$\ \mathbb{F}_{2^8}^{k\times k}$	$h \leftarrow\!\!\$\ \{p \in \mathbb{F}_{2^8}[x] \mid \deg(p) = d \text{ and } p(A) \neq 0\}$
$f \leftarrow\!\!\$\ \{p \in \mathbb{F}_{2^8}[x] \mid \deg(p) = d \text{ and } p(A) \neq 0\}$	$m_h \leftarrow\!\!\$\ [2, u] \cap \mathbb{Z}$
$m_f \leftarrow\!\!\$\ [2, u] \cap \mathbb{Z}$	$n_h \leftarrow\!\!\$\ [2, u] \cap \mathbb{Z}$
$n_f \leftarrow\!\!\$\ [2, u] \cap \mathbb{Z}$	$c_1 \leftarrow h(A)^{m_h}\, B\, h(A)^{n_h}$
$r \leftarrow f(A)^{m_f}\, B\, f(A)^{n_f}$	$ss \leftarrow h(A)^{m_h}\, r\, h(A)^{n_h}$
$pk \leftarrow (A, B, r)$	$c_2 \leftarrow H(ss) \oplus m$
$sk \leftarrow (f, m_f, n_f)$	$c \leftarrow (c_1, c_2)$
return pk, sk	**return** c

Dec(sk, c)
$(f, m_f, n_f) \leftarrow sk$
$(c_1, c_2) \leftarrow c$
$ss' \leftarrow f(A)^{m_f}\, c_1\, f(A)^{n_f}$
$m' \leftarrow H(ss') \oplus c_2$
return m'

Fig. 1. The PKE scheme defined in Definition 3.

– *A deterministic algorithm $Dec(sk, c)$ that takes a secret key sk and a ciphertext c and outputs a message m'.*

Collectively, these three algorithms form the PKE scheme proposed in [Hec20b]. The slightly different scheme proposed in [Hec20a] is obtained by requiring $m_h = m_f$ and $n_h = n_f$ during encryption (cf. Figure 1), with (m_f, n_f) being added to the public key.

We note that while no explicit reductions from breaking the schemes in [Hec20a,Hec20b] to solving the GSDP problem are given, in [Hec20a, § 4.3] it is claimed that Theorem 3 of [CDW07] implies IND-CPA security for the scheme in [Hec20a] given that the GSDP instance is hard. Yet, our attack will break the scheme without technically solving GSDP. In paticular, using the notation from Definition 2, we will recover $\alpha\, z^m$ and $\alpha^{-1}\, z^n$ for some non-zero $\alpha \in \mathbb{F}_{2^8}$, but not z itself.

Lemma 1 (Correctness of HK17-like PKE). *Let pp be public parameters and let $(pk, sk) \leftarrow\!\!\$\ Gen(pp)$ be a key pair for the scheme defined in Definition 3. Then for any $m \in \mathcal{M}$, $Dec(sk, Enc(pk, m)) = m$.*

Proof (Proof (sketch)). We start by noticing that given any ring R and matrix $A \in M_k(R)$, A commutes with the identity matrix I_k and with itself. This implies by direct computation that given any two polynomials $f, h \in R[x]$, $f(A)$ and $h(A)$ commute. From inspection of Figure 1 correctness of the scheme then follows, since

$$ss' = f(A)^{m_f}\, h(A)^{m_h}\, B\, h(A)^{n_h}\, f(A)^{n_f} = h(A)^{m_h}\, f(A)^{m_f}\, B\, f(A)^{n_f}\, h(A)^{n_h} = ss,$$

for any h, f, m_f, n_f, m_h, n_h and A.

In [Hec20a,Hec20b] the authors propose a cryptanalysis based on exhaustively guessing the secret polynomial f, increasing its degree to achieve security. As we will see, the presence of f does not provide any security, with the cost of the attack depending on the cost of linear algebra in $M_k(\mathbb{F}_{2^8})$ instead.

3.2 A warm-up attack

We first describe a bug in the implementation of the PKE schemes provided in [Hec20a,Hec20b] which results in a trivial attack. While the resulting attack is not particularly interesting, it does

serve as a warm-up for a technique used as a part of the linearisation attack against the correctly implemented scheme.

In [Hec20a,Hec20b], two similar implementations of the PKE in Wolfram Mathematica are provided. These are almost correct, except for a crucial bug during the computation of the public key share r, the ciphertext component c_1 and the shared secret ss, where rather than using matrix multiplication the authors use entry-wise matrix multiplication, which we now denote $\odot$. Being multiplication in $\mathbb{F}_{2^8}$ associative and commutative, so is entry-wise multiplication in $M_k(\mathbb{F}_{2^8})$. Given a public key share r of the form $r = f(A)^{m_f} \odot B \odot f(A)^{n_f}$ and a ciphertext component c_1 of the form $c_1 = h(A)^{m_h} \odot B \odot h(A)^{n_h}$, we can trivially compute

$$\begin{aligned} r \odot c_1 &= (f(A)^{m_f} \odot B \odot f(A)^{n_f}) \odot (h(A)^{m_h} \odot B \odot h(A)^{n_h}) \\ &= (h(A)^{m_h} \odot f(A)^{m_f} \odot B \odot f(A)^{n_f} \odot h(A)^{n_h}) \odot B \\ &= ss \odot B. \end{aligned}$$

We now define a "pseudoinverse" of a matrix with respect to entry-wise multiplication.

Definition 4. *Let K be a field and $M \in K^{r \times c}$ be a matrix with entries over K. We define the* pseudoinverse of M with respect to entry-wise multiplication *as the matrix M^+ with coefficients*

$$(M^+)_{i,j} \coloneqq \begin{cases} (M)_{i,j}^{-1} & \text{if } (M)_{i,j} \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, we can use the pseudoinverse B^+ of B to compute

$$ss \odot B \odot B^+ = \begin{cases} (ss)_{i,j}(B)_{i,j}(B)_{i,j}^{-1} & \text{if } (B)_{i,j} \neq 0, \\ 0 & \text{otherwise} \end{cases} = \begin{cases} (ss)_{i,j} & \text{if } (B)_{i,j} \neq 0, \\ 0 & \text{otherwise} \end{cases} = ss$$

by noticing that if $(B)_{i,j} = 0$ then $(ss)_{i,j} = 0$ too.

While this attack trivially breaks the examples provided in [Hec20a,Hec20b] and generated using the buggy code, it is clear from the analysis (cf. [Hec20b, § 4.2]) that r, c_1 and ss should be computed using matrix multiplication.

3.3 Linearisation attack

We consider an attack scenario where a passive eavesdropper intercepts a valid public key pk and a ciphertext $c \leftarrow Enc(pk, m)$ for some unknown message m, and tries to recover m. We consider three alternative approaches to the attack. We start by looking at the simplest one, which works whenever the secret matrix $f(A)$ is invertible, which for uniformly random A and f happens with very high probability. We then describe two alternatives to deal with increasingly tricky (and rare!) settings. We note that the third approach (described under "Case 2.2") is quite general, and solves all instances from our experiments, albeit at the cost of slightly higher runtime and implementation complexity.

Recall, given $A, B, r, c_1 \in M_k(\mathbb{F}_{2^8})$ such that $r = f(A)^{m_f}\, B\, f(A)^{n_f}$ and $c_1 = h(A)^{m_h}\, B\, h(A)^{n_h}$, we want to recover $ss = f(A)^{m_f}\, c_1\, f(A)^{n_f}$.

Case 1, $\det(f(A)) \neq 0$. The attack proceeds similarly to [BCM12] or [MSU11, § 4.4.1], by setting up the following linear system of equations in $2k^2$ unknowns in the form of two matrices $X, Y \in \mathbb{F}_{2^8}^{k \times k}$:

$$r\, X = Y\, B, \tag{1}$$
$$A\, X = X\, A, \tag{2}$$
$$A\, Y = Y\, A. \tag{3}$$

By construction, we know that the system has non-zero solutions $X = \alpha f(A)^{-n_f}$ and $Y = \alpha f(A)^{m_f}$ for any $\alpha \in \mathbb{F}_{2^8} \backslash \{0\}$. Once a solution X, Y is found, the adversary derives a guess zz for

the value of ss', $zz \coloneqq Y\, c_1\, X^{-1}$. By (2) and (3), we know that X^{-1} and Y will commute with any polynomial in A, while by (1) we know that $r = Y\, B\, X^{-1}$, implying $zz = ss$:

$$\begin{aligned} zz &= Y\, c_1\, X^{-1} \\ &= Y\, h(A)^{m_h}\, B\, h(A)^{n_h}\, X^{-1} \\ &= h(A)^{m_h}\, Y\, B\, X^{-1}\, h(A)^{n_h} && \text{by (2) and (3),} \\ &= h(A)^{m_h}\, r\, h(A)^{n_h} && \text{by (1),} \\ &= ss. \end{aligned}$$

Case 2, $\det(f(A)) = 0$. This case is at first glance trickier. Indeed, $\det(f(A)) = 0 \Leftrightarrow \det(f(A)^{n_f}) = 0 \Leftrightarrow \det(f(A)^{m_f}) = 0$, meaning that the system described above will not have non-zero solutions $X = \alpha f(A)^{-n_f}$, and inverting the roles of X and Y so that (1) became $Yr = BX$ (and so Y yielded $\alpha f(A)^{-m_f}$) would similarly not work.

Let $p_A(x)$ be the characteristic polynomial of A and $K \supset \mathbb{F}_{2^s}$ be the splitting field of p_A. We consider two further subcases of this problem, based on whether A is diagonalisable over K or not, to recover the shared secret in this case. We will now abuse notation and identify A with its canonical lift into K.

Case 2.1, diagonalisable A. Suppose A (seen as a matrix in $K^{k\times k}$) is diagonalisable, such that $A = P\, D\, P^{-1}$, with D diagonal and P invertible. It follows that $f(A)^t = P\, f(D)^t\, P^{-1}$, for any non-negative integer t. We can then rewrite the public key component r as

$$\begin{aligned} r &= f(A)^{m_f}\, B\, f(A)^{n_f} \\ &= P\, f(D)^{m_f}\, P^{-1}\, B\, P\, f(D)^{n_f}\, P^{-1} \\ &\iff P^{-1} r P = f(D)^{m_f}\, (P^{-1}\, B\, P)\, f(D)^{n_f} \\ &\iff \hat{r} = f(D)^{m_f}\, \hat{B}\, f(D)^{n_f}, \end{aligned} \tag{4}$$

by defining $\hat{r} \coloneqq P^{-1} r P$ and $\hat{B} = P^{-1}\, B\, P$.

Without loss of generality, assume that

$$f(D) = \begin{bmatrix} f_1 & & & & & \\ & \ddots & & & & \\ & & f_s & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix}$$

where the $f_1, \ldots, f_s$ are all non-zero. Recalling Definition 4 from the "warm-up attack" in Section 3.2, we can rewrite (4) as

$$\begin{aligned} \hat{r}\, f(D)^{n_f +} &= f(D)^{m_f}\, \hat{B}\, f(D)^{n_f}\, f(D)^{n_f +} \\ &= f(D)^{m_f}\, \hat{B}\, I_{n;s} \\ &= f(D)^{m_f}\, \bar{B}, \end{aligned}$$

$$\text{where} \quad (I_{n;s})_{i,j} = \begin{cases} 1 & \text{if } (f(D))_{i,j} \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \bar{B} \coloneqq \hat{B}\, I_{n;s},$$

where we estimate the indices where $(f(D))_{i,j} \neq 0$ by noting that j must equal i, and that whenever $(f(D))_{i,i} = 0$, the i^{th} row and column of $\hat{r}$ are zeroed.

We can then set up a linear system of equations over K in $2s$ unknowns that for ease of exposition we represent in the form of two matrices X, $Y \in K^{k\times k}$, such that

$$\hat{r}\, X = Y\, \bar{B}, \tag{5}$$

$$(X)_{i,j} = (Y)_{i,j} = 0 \ \forall i \neq j, \tag{6}$$

$$(X)_{i,i} = (Y)_{i,i} = 0 \ \forall i > s, \tag{7}$$

where (5) has the same role as (1) in Case 1, and (6) and (7) enforce the expected shape of $f(D)$. Experimentally, we verify that solutions yield $X = \alpha f(D)^{n_f{}^+}$ and $Y = \alpha f(D)^{m_f}$, which leads to recovery of $\alpha\, f(A)^{m_f} = P\, Y\, P^{-1}$ and $\alpha^{-1}\, f(A)^{n_f} = P\, X^+\, P^{-1}$, for some non-zero $\alpha \in \mathbb{F}_{2^8}$. We then define recover the shared secret by computing $\alpha f(A)^{m_f}\, c_1\, \alpha^{-1} f(A)^{n_f} = ss$.

Case 2.2, non-diagonalisable A. In our experiments, Cases 1 and 2.1 cover the vast majority of the random instances generated. However, some outliers where A is not diagonalisable and $\det(f(A)) = 0$ remain. To cover this case, we extend the Case 2.1 attack. The resulting process is general and solves all instances of the problem we have generated throughout experiments, including those falling in Cases 1 and 2.1. However, our implementation of it is slightly slower than those of the other cases. Hence, we only use the following attack when the other approaches fail.

The idea of Case 2.2 is to replace diagonalisation with Jodan Canonical Form (JCF) computation. Let $J \in K^{k\times k}$ be the JCF of A over K, such that $A = P\, J\, P^{-1}$ and $f(A)^t = P\, f(J)^t\, P^{-1}$ for any non-negative integer t. As done in Case 2.1, we rewrite

$$r = f(A)^{m_f}\, B\, f(A)^{n_f} \iff \hat{r} = f(J)^{m_f}\, \hat{B}\, f(J)^{n_f},$$

by defining $\hat{r} \coloneqq P^{-1} r P$ and $\hat{B} = P^{-1}\, B\, P$.

By definition, J is block-diagonal, and so are its powers. Therefore, $f(J)$ is block-diagonal itself by being a sum of block-diagonal matrices, and so are non-negative powers of $f(J)$,

$$f(J) = \begin{bmatrix} \boxed{f_1} & & \\ & \ddots & \\ & & \boxed{f_s} \end{bmatrix}.$$

We adapt Definition 4 of a pseudoinverse diagonal matrix to the setting of block-diagonal matrices, by constructing a block-diagonal matrix $f(J)^{\oplus}$ where we invert the non-singular blocks, and set to zero the singular ones. The corresponding product $I_{f(J)} \coloneqq f(J)\, f(J)^{\oplus}$ is a matrix with ones on the diagonal where a non-singular block was, and zeroes otherwise, similar to $I_{n;s}$ in Case 2.1. Following the same approach as in Case 2.1, we define $\bar{B} \coloneqq \hat{B} I_{f(J)}$ where we compute $I_{f(J)}$ by looking at the indices i such that the i^{th} row and column of $\hat{r}$ are zeroed. We then set a linear system with the following constraints:

$$\hat{r}\, X = Y\, \bar{B}, \tag{8}$$

$$X \text{ and } Y \text{ be block-diagonal with } J(A)\text{'s structure}, \tag{9}$$

$$J\, X = X\, J \tag{10}$$

$$J\, Y = Y\, J, \tag{11}$$

our intention being that X yields a solution of the form $\alpha f(J)^{n_f{}^{\oplus}}$ and Y yields $\alpha f(J)^{m_f}$. We note that (9) is a set of linear constraints that force the lower-triangular indices of X and Y as well as the top-triangular ones not falling inside one of the Jordan blocks, to be zero. We also reintroduce commutativity constraints for the solutions, but with J rather than A (since $f(A)A = Af(A) \iff f(J)J = Jf(J)$). This is because block-diagonal matrices don't commute in general. The requirement was absent in Case 2.1 since diagonal matrices do. Experimentally, we verify that these steps result in solutions for the few cases with singular $f(A)$ and non-diagonalisable A.

Piecing the attack together. In our attack implementation, we proceed as follows. If $\det(r) \neq 0$ we know that $f(A)$ is invertible, and hence use the Case 1 steps. If $\det(r) = 0$ but A is diagonalisable, we attempt the Case 2.1 steps. We note that we may end up using these steps on an instance where $f(A)$ is invertible but B isn't. While we can test for invertibility of B, we cannot for $f(A)$ since it would require guessing f. However, in practice the steps lead to a solution with high probability regardless of the invertibility of $f(A)$. Finally, if $\det(r) = 0$ and A is not diagonalisable, we use the Case 2.2 steps.

k	d	$\log_2 u$	# tries	p_{succ}	avg. runtime (s)	attack case		
						1	2.1	2.2
2	7	32	200	1.0	0.02	197	3	0
2	15	32	200	1.0	0.02	197	3	0
2	23	32	200	1.0	0.02	199	1	0
2	31	32	200	1.0	0.02	200	0	0
2	7	64	200	1.0	0.02	199	1	0
2	15	64	200	1.0	0.02	200	0	0
2	23	64	200	1.0	0.02	200	0	0
2	31	64	200	1.0	0.02	198	2	0
3	7	32	200	1.0	0.03	198	2	0
3	15	32	200	1.0	0.03	196	4	0
3	23	32	200	1.0	0.03	199	1	0
3	31	32	200	1.0	0.03	197	3	0
3	7	64	200	1.0	0.03	199	1	0
3	15	64	200	1.0	0.03	197	3	0
3	23	64	200	1.0	0.03	197	2	1
3	31	64	200	1.0	0.03	200	0	0
10	40	64	20	1.0	0.67	20	0	0
15	40	64	20	1.0	2.51	20	0	0
20	40	64	20	1.0	6.92	20	0	0
24	40	64	20	1.0	12.93	20	0	0
100	40	64	2	1.0	4527.11	2	0	0

Table 1. Benchmarks for the attack on different parameter sets. k is the matrix rank, d is the polynomial degree, u is the upper bound for the polynomial exponents. Each set of parameters is run with "# trials" different PRNG seeds. We average the runtime of the attack (in seconds) and report the measured success probability p_{succ}.

Remark 3. We note that while on one hand the adversary does not recover the secret key f, on the other she does not need it to decrypt arbitrary ciphertexts. As a consequence, the attack cannot be stopped by picking f of a higher degree.

Remark 4. In [Hec20a], the authors mention that the values r and c_1 would need to be "disguised" in order to avoid the possibility of an adversary recovering of the powers of $f(A)$ and $h(A)$ by inspection of r or c_1. However, no concrete disguising mechanism is given. The attack we describe does not inspect in any specific way r or c_1, only using their entries as matrices in $\mathbb{F}_{2^8}$ to set up a linear system of equations. Since any party encrypting a message needs to compute ss by evaluating matrix multiplications with r, we suspect any disguise would require keeping r as some form of $k \times k$ matrix, making the attack possible.

3.4 Attack implementation

We implement the scheme and the attack in Sagemath version 9.4 [S+19] and run it on a single Intel(R) Core(TM) i7-4790K CPU clocked at 4.00GHz on various different parameter sets. We note that the attack covers also the case where an extra condition $m_f = m_h$ and $n_f = n_h$ is imposed as in [Hec20a], since recovery of $\alpha f(A)^{m_f}$ and $\alpha^{-1} f(A)^{n_f}$ for some non-zero $\alpha \in \mathbb{F}_{2^8}$ only requires the public key, which is not affected by such condition. In Table 1, we report parameter sets and measured runtimes of the attack. The source for replicating the attack can be found at https://github.com/fvirdia/popping-r-propping-code-and-data.

Results. The results of our experiments can be found in Table 1. Experimentally, we observe the attack succeed with probability 1 for all the parameter sets attempted. In the case of 2×2 and 3×3 matrices (the ranks suggested in [Hec20a,Hec20b]), the attack takes negligible time. In the case of the two attempted 100×100 matrices, the attack takes less than 2 hours on a single core.

Gen(pp)

1 $g_0, |G| \leftarrow pp$
2 $r \leftarrow\!\!\$ \{2, \ldots, |G| - 2\}$
3 $a \leftarrow\!\!\$ \{2, \ldots, |G| - 2\}$
4 $g \leftarrow g_0^r$
5 $pk \leftarrow g^a$ // Note $g^a = g_0^{a \cdot r}$
6 $sk \leftarrow (r,\, g^a)$
7 **return** pk, sk

Forge(pk, m)

1 $h \leftarrow H(m)$
2 $t \leftarrow$ any value in $\mathbb{Z}$
3 $s_2 \leftarrow pk^t$
4 $s_1 \leftarrow pk\, s_2^{-h}$
5 $\sigma \leftarrow (s_1, s_2)$
6 **return** σ

Sign(pp, sk, m)

1 $g_0, |G| \leftarrow pp$
2 $r,\, g^a \leftarrow sk$
3 $g \leftarrow g_0^r$
4 $h \leftarrow H(m)$
5 $n \leftarrow\!\!\$ \{2, \ldots, |G| - 2\}$
6 $s_1 \leftarrow g^a \cdot (g^{-1})^{n \cdot h}$
7 $s_2 \leftarrow g^n$
8 $\sigma \leftarrow (s_1, s_2)$ // Note $s_1 = pk \cdot (s_2)^{-h}$
9 **return** σ

Vrfy(pk, σ, m')

1 $s_1,\, s_2 \leftarrow \sigma$
2 $h' \leftarrow H(m')$
3 **if** $s_1\, s_2^{h'} = pk$
4 **return** `true`
5 **else return** `false`

Fig. 2. The signature scheme from [Hec21a], desribed using the syntax from Definition 5. We also include an algorithm to forge signatures.

4 Forgery attack

In [Hec21a], the authors propose a signature scheme based on the hardness of the Discrete Logarithm Problem (DLP) over $M_k(\mathbb{F}_{2^8})$. While we will discuss the practical hardness of such DLP instances in Section 5, in this section we describe a forgery attack on the signature scheme that is independent of the hardness of the DLP.

Definition 5. *Let k be a positive integer, let $g_0 \in GL_k(\mathbb{F}_{2^8})$ be an invertible matrix and let G be the group generated by g_0. Let $pp = (g_0, |G|)$ be a tuple of public parameters. Let $\mathcal{M}$ be the message space, and let $H : \mathcal{M} \to \mathbb{Z}_{|G|}$ be a random oracle. In Figure 2 we define three algorithms:*

- *A probabilistic algorithm Gen(pp) that takes public parameters pp and outputs a public and private key pair (pk, sk).*
- *A probabilistic algorithm Sign(pp, sk, pk, m) that takes public parameters pp, a secret key sk and a message m and outputs a signature σ.*
- *A deterministic algorithm Vrfy(pk, σ, m') that takes a public key pk, a signature σ and a message m' and outputs a boolean value.*

Remark 5. All elements in $M_k(\mathbb{F}_{2^8})$ computed as part of the key generation and signature algorithms are powers of the public parameter g_0. Therefore, although $M_k(\mathbb{F}_{2^8})$ is non-commutative, operations effectively happen inside the commutative multiplicative subgroup $G \subset M_k(\mathbb{F}_{2^8})$.

From the Vrfy algorithm in Figure 2, we can see that the verification check is simply checking that the two group elements making the signature σ multiply to pk. Forging such elements is trivial since powers and inversions in $M_k(\mathbb{F}_{2^8})$ are easy to compute. In Figure 2 we provide a Forge algorithm that given any message and public key, forges a signature. We notice that although not mandated by the Vrfy algorithm, signature components are always a power of g. Therefore, we reuse the public key pk such that forged signature components are also powers of g.

We remark that the signature algorithm proposed in [Hec21a] differs significantly from ElGamal's signature scheme [ElG85], and these differences allow forgeries to be created.

5 Solving the Discrete Logarithm Problem over $M_k(\mathbb{F}_{2^8})$

The Discrete Logarithm Problem (DLP) has a rich cryptanalysis history. In our case, a variant of the DLP using matrices in $M_k(\mathbb{F}_{2^8})$ was proposed to instantiate signatures [Hec21a], group

k	Claimed classical security (bits)	Claimed quantum security (bits)	# tries	Average runtime (s)	Measured Pr[success]
3	24	12	128	0.017	1.0
4	32	16	128	0.023	1.0
7	96	48	128	0.048	1.0
10	112	56	128	0.126	1.0
12	160	80	128	0.388	1.0

Table 2. Claimed classical and quantum bit-hardness of the DLP over specific subgroups of $M_k(\mathbb{F}_{2^8})$ proposed in [Hec21c,Hec21a,Hec21b], and average measured runtime for solving random DLP instances using the suggested values for g_0.

key exchange [Hec21c] and oblivious transfer [Hec21b]. For all three proposals, the authors claim post-quantum security and provide particular instances of subgroup generators to be used for achieving given security levels. These are summarised in Table 2.

We note however that "R-propping" the DLP does not achieve post-quantum security. Indeed, Menezes and Wu [MW97] show a probabilistic polynomial time reduction of the DLP over the group $\mathrm{GL}_k(\mathbb{F}_q)$ of invertible matrices over $\mathbb{F}_q$ to k^2 instances of the DLP over some extensions fields of $\mathbb{F}_q$. This reduction is further extended to the group of all matrices $M_k(\mathbb{F}_q)$ in [Mya]. These reductions imply quantum polynomial-time attacks using Shor's algorithm [Sho94].

Looking further, a direct implementation of Menezes and Wu's reduction demonstrates that for the proposed generators the order of the extension fields of $\mathbb{F}_{2^8}$ stays relatively small, and well below the size of the record DLP computations carried over binary fields [BBD+14,GGMZ13,GKZ14,GKZ]. We demonstrate this by breaking various DLP instances set using the generators in $M_k(\mathbb{F}_{2^8})$ proposed for "R-propping" schemes, for every suggested dimension $k = 3$, 4, 7, 10, 12.

5.1 Attack implementation

We do a direct implementation of Menezes and Wu [MW97, Algorithm 2] using Sagemath 9.4, to reduce the DLP problem over $M_k(\mathbb{F}_{2^8})$ into a set of DLP problems over extensions of $\mathbb{F}_{2^8}$. We then solve these instances using Sagemath's default discrete logarithm routine. We note that while the proof of Menezes and Wu indicates specific algorithms for computing characteristic polynomials and Jordan Canonical Forms in order to formally argue probabilistic polynomial time, we use Sagemath's default routines for both these steps. We also notice that for all the challenges attempted, the value of t in Step 4 of [MW97, Algorithm 2] is always equal to 1, meaning that we can skip the step entirely (which therefore we don't implement).

All our experiments are run on a single Intel(R) Core(TM) i7-4790K CPU clocked at 4.00GHz. For every generator suggested in [Hec21c,Hec21a,Hec21b], we generate 128 random exponents and solve the corresponding DLP instances. Every attempted instance is correctly solved. On average the largest instances take less than 0.5 seconds to solve. We report our results in Table 2. The source for replicating the attack can be found at https://github.com/fvirdia/popping-r-propping-code-and-data.

As part of our experiments, we also note that the order of the proposed generators for $k = 7$ is $\approx 2^{48}$ rather than $2^{96} - 1$, as reported in [Hec21c,Hec21a,Hec21b]. Similarly, for $k = 10$ the order is $\approx 2^{64}$ rather than $2^{112} - 1$, and for $k = 12$ is $2^{80} - 1$ rather than $2^{160} - 1$.

Acknowledgements. This work was supported by the Zurich Information Security and Privacy Center. We thank the reviewers at *Mathematical Cryptology* for their useful comments.

References

[AES01] Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, November 2001.

[BBD09] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors. *Post-Quantum Cryptography*. Springer Berlin Heidelberg, 2009.

[BBD+14] Razvan Barbulescu, Cyril Bouvier, Jérémie Detrey, Pierrick Gaudry, Hamza Jeljeli, Emmanuel Thomé, Marion Videau, and Paul Zimmermann. Discrete logarithm in GF(2^{809}) with FFS. In *International Workshop on Public Key Cryptography*, pages 221–238. Springer, 2014.

[BCM12] Simon R. Blackburn, Carlos Cid, and Ciaran Mullan. Cryptanalysis of three matrix-based key establishment protocols. *Journal of Mathematical Cryptology*, 5(2):159–168, 2012.

[BD94] Mike Burmester and Yvo Desmedt. A secure and efficient conference key distribution system. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 275–286. Springer, 1994.

[CDW07] Zhenfu Cao, Xiaolei Dong, and Licheng Wang. New public key cryptosystems using polynomials over non-commutative rings. Cryptology ePrint Archive, Report 2007/009, 2007. https://eprint.iacr.org/2007/009.

[CO15] Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In *International Conference on Cryptology and Information Security in Latin America*, pages 40–58. Springer, 2015.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[GGMZ13] Faruk Göloğlu, Robert Granger, Gary McGuire, and Jens Zumbrägel. On the function field sieve and the impact of higher splitting probabilities. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 109–128, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[GKZ] Robert Granger, Thorsten Kleinjung, and Jens Zumbragel. Discrete logarithms in gf(2^{9234}). Accessed: 2022-07-22. Archived at https://web.archive.org/web/20220721215952/https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY%3B9aa2b043.1401.

[GKZ14] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. Breaking '128-bit secure' supersingular binary curves. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, pages 126–145, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[Hec20a] Pedro Hecht. PQC: R-propping of public-key cryptosystems using polynomials over non-commutative algebraic extension rings. Cryptology ePrint Archive, Report 2020/1102, 2020. https://eprint.iacr.org/2020/1102.

[Hec20b] Pedro Hecht. R-propping of HK17: Upgrade for a detached proposal of NIST PQC first round survey. Cryptology ePrint Archive, Report 2020/1217, 2020. https://eprint.iacr.org/2020/1217.

[Hec21a] Pedro Hecht. PQC: R-propping of a new group-based digital signature. Cryptology ePrint Archive, Report 2021/270, 2021. https://eprint.iacr.org/2021/270.

[Hec21b] Pedro Hecht. PQC: R-propping of a simple oblivious transfer. Cryptology ePrint Archive, Report 2021/854, 2021. https://eprint.iacr.org/2021/854.

[Hec21c] Pedro Hecht. PQC: R-propping of burmester-desmedt conference key distribution system. Cryptology ePrint Archive, Report 2021/024, 2021. https://eprint.iacr.org/2021/024.

[HK17] Juan Pedro Hecht and Jorge Alejandro Kamlofsky. HK17. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[MSU11] Alexei Myasnikov, Vladimir Shpilrain, and Alexander Ushakov. *Non-Commutative Cryptography and Complexity of Group-Theoretic Problems*. American Mathematical Society, USA, 2011.

[MW97] Alfred J Menezes and Yi-Hong Wu. The discrete logarithm problem in gl (n, q). *Ars Combinatoria*, 47:23–32, 1997.

[Mya] Alex Myasnikov. Discrete logarithm problem for matrices over finite group rings. https://web.stevens.edu/algebraic/GAGTA/Slides/Myasnikov.pdf.

[S+19] William Stein et al. *Sage Mathematics Software Version 9.4*. The Sage Development Team, 2019. http://www.sagemath.org.

[Sho94] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.