New Public-Key Cryptosystem Blueprints Using Matrix Products in $\mathbb{F}_p$

Remi Geraud-Stewart[2,⋆,[0000−0001−8719−1724]] and David Naccache[1]

[1] DIÉNS, ÉNS, CNRS, PSL University, Paris, France
david.naccache@ens.fr
[2] Qualcomm Inc., San Diego, USA
rgerauds@qti.qualcomm.com

Abstract. Given a set of matrices $\mathbf{A} := \{A_0, \ldots, A_{k-1}\}$, and a matrix M guaranteed to be the product of some ordered subset of $\mathbf{L} \subset \mathbf{A}$, can $\mathbf{L}$ be efficiently recovered? We begin by observing that the answer is positive under some assumptions on $\mathbf{A}$.
Noting that appropriate transformations *seem* to make $\mathbf{L}$'s recovery difficult we provide the blueprint of two new public-key cryptosystems based upon this problem.
We term those constructions "blueprints" because, given their novelty, we are still uncertain of their exact security. Yet, we daringly conjecture that even if attacks are found on the proposed constructions, these attacks could be thwarted by adjustments in the key generation, key size or the encryption mechanism, thereby resulting on the long run in fully-fledged public-key cryptosystems that do not seem to belong to any of the mainstream public-key encryption paradigms known to date.

1 Introduction

There are relatively few known public-key cryptosystem families, and despite multiple efforts such schemes have largely resisted classification attempts. While interest in novel cryptosystems was recently reignited by progress in quantum computing, most alternative constructions only provide key encapsulation mechanisms (KEMs), which are sufficient in many practical settings as substitutes to key exchange. Such constructions involve for instance lattices [Pei15], isogenies [SHS21], or even plactic monoids [Bro21, Mon22].

Still the challenge remains to identify less restricted structures upon which the most general public-key encryption paradigms can be built. This paper introduces new public-key encryption mechanisms that seem to have been overlooked.

A word of caution: We term the constructions given here "blueprints" as, given their novelty, we are uncertain of their exact security. Yet, we daringly conjecture

⋆ The bulk of this work was performed when this author was with ÉNS.

that even if attacks are to be found on the proposed constructions in the future, these attacks could be thwarted by proper modifications in the key generation or in the encryption mechanism, thereby resulting on the long run in fully-fledged public-key cryptosystems that do not seem to belong to any of the mainstream public-key encryption family known to date. Part of the reason of our shyness here is that we are well aware of *many* broken attempts at non-commutative cryptography (see, e.g., the review in [PQ11] or [HS03]). We were made aware of a possibly similar construction in concurrent work [PLZG23, Sec. 3]; it seems that this is a special case of the ideas discussed here.

1.1 The intuition

Fix a finite field $\mathbb{F}_p$ of odd prime order, a positive integer n, and consider $n \times n$ matrices with entries in $\mathbb{F}_p$. Fix a set of such matrices $\mathbf{A} := \{A_0, \ldots, A_{k-1}\}$. For every permutation σ of the set $\{0, \ldots, k-1\}$ let:

$$\sigma\mathbf{A} := \prod_{i=0}^{k-1} A_{\sigma(i)}.$$

Assuming that every choice of σ yields a different matrix $\sigma\mathbf{A}$ (i.e., that $\sigma \mapsto \sigma\mathbf{A}$ is injective), can we *efficiently* recover σ given $\mathbf{A}$ and $\sigma\mathbf{A}$? We refer to this problem as the *decomposition* of $\sigma\mathbf{A}$ over $\mathbf{A}$.

How hard the decomposition problem is depends on the choice of p, n, k, and $\mathbf{A}$. In the worst case $k!$ products are to be considered, which makes exhaustive search intractable even for small k (e.g., $24! \simeq 2^{79}$). Indeed, matrix product is in general non-commutative and therefore order matters.

However for appropriate choices of $\mathbf{A}$ we propose a polynomial-time algorithm[3] solving the decomposition problem (see Section 2 for details).

Interestingly, it appears that a simple reversible transformation of $\mathbf{A}$ (all other parameters being fixed) into a new set $\overline{\mathbf{A}}$ makes *that* algorithm fail (see Section 3 for details). This yields the following two cryptosystem blueprints:

1. *Direct decomposition.* Define a set of matrices $\mathbf{A}$ upon which decomposition is easy, and transform $\mathbf{A}$ into $\overline{\mathbf{A}}$ over which decomposition is hard. The public key is $\overline{\mathbf{A}}$. Map the plaintext to a permutation σ and define $\sigma\overline{\mathbf{A}}$ as the ciphertext. To decrypt, transform $\sigma\overline{\mathbf{A}}$ back into a $\sigma\mathbf{A}$ which is easy to decompose over $\mathbf{A}$ thereby yielding the message σ.
2. *Alternating decomposition.* Define *two* sets of matrices $\mathbf{A}^b = \{A_0^b, \ldots, A_{k-1}^b\}$ with $b \in \{0, 1\}$ on which decomposition is easy[4]. Transform $\mathbf{A}^b$ into public-key

[3] $O(k^2 n^3 \log^2 p)$ assuming no sophisticated asymptotic optimizations such as [AW20].
[4] It is straightforward to extend the scheme to non-binary bases, e.g. $0 \leq b \leq \ell - 1$. The case $\ell = 2$ just makes our exposition simpler.

matrices $\overline{\mathbf{A}}^b$ over which decomposition is hard. Given a plaintext $m \in \{0,1\}^k$, form the ciphertext:

$$m\overline{\mathbf{A}} := \prod_{i=0}^{k-1} \overline{A}_i^{m_i},$$

where $\mathbf{A} := (\mathbf{A}^0, \mathbf{A}^1)$ and $\overline{\mathbf{A}} := (\overline{\mathbf{A}}^0, \overline{\mathbf{A}}^1)$. To decrypt, use the reverse transformation to obtain $m\mathbf{A}$, from which m is recovered.

To the best of our knowledge, these constructions do not belong to any of the well-established public-key cryptosystem families known to date.

1.2 Structure of this paper

The rest of this paper is structured as follows. Section 2 provides details on the decomposition algorithm and its complexity. Section 3 describes the trapdoor allowing to switch between easy and hard decomposition instances. Section 4 describes both cryptosystems in their bare, deterministic form. Section 5 discusses security and suggests concrete implementation parameters. Finally, Section 6 discusses some questions requiring further scrutiny regarding the security of our constructions.

2 Decomposition algorithm

A number of clarifications are necessary to introduce the decomposition algorithm mentioned in the introduction. Consider a set $\mathbf{A}$ of non-singular matrices.

To illustrate the *idea*, consider three matrices A_0, A_1, A_2 over $\mathbb{Q}$, with positive integer entries. The product A_0A_1 has positive integer entries, which cannot be smaller than those of A_0 of A_1. In general, the inverses $A_0^{-1}, A_1^{-1}, A_2^{-1}$ do *not* have positive integer entries — this is easily seen using the adjoined formula for matrix inversion. Therefore, in general, a product of the form $A_i^{-1}M$ does not have positive integer entries, even though M does. This leads to the following observation: if we assume that M is the product of A_0, A_1, A_2 in some order, then with high probability *only one* of the products $A_i^{-1}M$ will feature integer entries. For instance if $M = A_1A_2A_3$ then only $A_1^{-1}M$ will succeed. In addition, in the very rare event where, following a bad guess, entries will result in integers, then their *sizes* will betray the bad guess.

Our constructions will not use matrices over $\mathbb{Q}$, because there appears to be no set of matrices on which decomposition is hard over $\mathbb{Q}$. We make the following assumption[5]:

[5] This conjecture is supported both by theoretical results [BG08, BS92, BKT04] and empirical evidence.

Conjecture 1. Fix a security parameter λ. There exist values p, k, n polynomial in λ such that, if $\mathbf{A} \subset \mathrm{GL}_n(\mathbb{F}_p)$ of size k is sampled uniformly without replacement and σ is a uniformly sampled permutation, no probabilistic algorithm can solve the decomposition problem of $\sigma\mathbf{A}$ over $\mathbf{A}$ with non-negligible probability in time polynomial in λ.

To translate the intuition underlying the decomposition algorithm over finite fields, we need a notion of "size" for field elements. This is why we restrict ourselves to *prime* finite fields[6]. The size of an element $x \in \mathbb{F}_p$ shall be its representation as an integer in $\{0, \ldots, p-1\}$. We denote the size of x by $|x|$.

We extend this notion entry-wise to matrices and write $M_1 \leq M_2$ if $|(M_1)_{i,j}| \leq |(M_2)_{i,j}|$ for all i, j. Fix a parameter $0 < \alpha < p$. We will refer to matrices M such that $|M| \leq \alpha$ as *dwarves*; in contrast, unconstrained matrices over $\mathbb{F}_p$ will be referred to as *elves*[7]. Formally, let:

$$\begin{aligned} \mathfrak{E} &:= \mathrm{GL}_n(\mathbb{F}_p) - Z(\mathrm{GL}_n(\mathbb{F}_p)), \\ \mathfrak{D} &:= \{M \in \mathfrak{E} \text{ such that } |M| \leq \alpha\}, \end{aligned}$$

respectively the set of elves and of dwarves. We exclude the center for technical reasons, as we know ahead of time that such matrices won't be usable.

Let $\mathbf{A} \subset \mathfrak{D}$. Then the largest coefficient in $\sigma\mathbf{A}$ does not exceed $n^{k-1}\alpha^k$. If $n^{k-1}\alpha^k$ doesn't exceed p, then we are in fact operating over the integers. Therefore we can use the above discussion about integers, materialized in the algorithm of Figure 1, to recover σ (see Remarks 1 and 6 for some precision about this).

Input: p prime, $n > 1$, M an $n \times n$ matrix over $\mathbb{F}_p$,
$\mathbf{A} = \{A_0, \ldots, A_{k-1}\}$ invertible matrices over $\mathbb{F}_p$
Output: $\{A_{i_0}, \ldots, A_{i_{k-1}}\}$ such that $M = \prod A_{i_j}$, or $\perp$

Decompose($\mathbf{A}, M$):

1. **if** $|\mathbf{A}| = 0$ **return** $\perp$.
2. **if** $|\mathbf{A}| = 1$ and $M \notin \mathbf{A}$ **return** $\perp$ **else return** $\mathbf{A}$.
3. **for** $A \in \mathbf{A}$,
   3.1 $M' \leftarrow A^{-1}M$.
   3.2 **if** $M' \leq M$,
      3.2.1. $\mathbf{L} \leftarrow$ Decompose($\mathbf{A} - \{A\}, M'$).
      3.2.2. **if** $\mathbf{L} = \perp$ **return** $\perp$ **else return** $\{A\} \cup \mathbf{L}$.
4. **return** $\perp$ // as no matrix worked.

Fig. 1. Polynomial-time decomposition algorithm over $\mathbf{A}$ in $\mathbb{F}_p$.

[6] Another reason to avoid field extensions is to avoid having to worry about subfields.
[7] Besides the allusion to size difference, as we shall see, dwarves are less affected by the ring than elves, which is consistent with prior work [Tol12, Part V].

However, if there is even one matrix in $\mathbf{A}$ that doesn't belong to $\mathfrak{D}$, then this algorithm is likely to fail. This motivates the following conjecture, which is well supported by empirical evidence:

Conjecture 2. There exist values n and k such that no efficient algorithm can solve the decomposition problem $\sigma\mathbf{A}$ over $\mathbb{F}_p$, for a uniformly sampled subset $\mathbf{A} \subset \mathfrak{E}$.

Remark 1 (Incorrect guesses and backtracking). Algorithm Decompose assumes that only one matrix can satisfy the constraint; but because we work in $\mathbb{F}_p$, there is a non-zero probability η that an incorrect candidate will still result in smaller coefficients by chance. Should this occur, it is likely that an incorrect guess would negatively affect the next decomposition step, resulting in *no* candidate being selected, and the algorithm returning $\perp$.

Fortunately, this rare situation is nicely handled by low-depth (and hence efficient) backtracking. If η is small enough, as is the case for well-chosen parameters, the overall algorithm remains polynomial-time. Note that η decreases rapidly at each algorithm step.

Remark 2 (Complexity and memory). If we ignore backtracking, assuming that elements in $\mathbf{A}$ are provided together with their inverses, and that Decompose completes without returning $\perp$, we can see that Decompose claims $k(k-1)/2$ matrix multiplications, and n^2 field element comparisons. The latter is negligible when compared to the former so Decompose's overall complexity is dominated by $O(k^2)$ matrix products over $\mathbb{F}_p$: with textbook algorithms this is a total of $O(k^2 n^3 \log^2 p)$ time. Using efficient data structures, Decompose uses at most k temporary matrices, and can reuse that memory to return the final list, totaling a memory footprint of $O(kn^2 \log p)$.

Remark 3 (Speed versus accuracy trade-off). We can quickly eliminate some candidates by checking only one matrix entry (or a few), rather than all entries. This increases η. Therefore the gains of this approach are to be balanced against the extra backtracking caused by it. Note however that such shortcuts leak information to an attacker knowing which matrix element is about to be tested by the receiver. This leakage may can be reduced by checking one or a few *random* entries at each round instead of a fixed one.

Example 1. Let $\sigma = (2, 1, 3)$ and $\{n, k, \alpha, p\} = \{2, 3, 3, 1234567891\}$. We have $\mathbf{A} = \{A_1, A_2, A_3\}$ with:

$$A_1 = \begin{pmatrix} 0 & 1 \\ 2 & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix}, \quad \text{and } \sigma\mathbf{A} = \begin{pmatrix} 11 & 6 \\ 13 & 8 \end{pmatrix}.$$

We can now compute $A_i^{-1} \times \sigma\mathbf{A}$ for $i = 1, 2, 3$ (decreasing entries are in blue, whereas increasing one are in red):

$$A_1^{-1}\sigma\mathbf{A} = \begin{pmatrix} 617283952 & 4 \\ 11 & 6 \end{pmatrix},$$
$$A_2^{-1}\sigma\mathbf{A} = \begin{pmatrix} 3 & 2 \\ 4 & 2 \end{pmatrix},$$
$$A_3^{-1}\sigma\mathbf{A} = \begin{pmatrix} 9 & 4 \\ 1234567884 & 1234567889 \end{pmatrix},$$

Only A_2 results in a matrix with *all* coefficients smaller. The only and best candidate is therefore A_2. Continuing, we multiply on the left by the inverses of A_1 and A_3:

$$A_1^{-1}\left(A_2^{-1}\sigma\mathbf{A}\right) = \begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix}, \quad A_3^{-1}\left(A_2^{-1}\sigma\mathbf{A}\right) = \begin{pmatrix} 2 & 2 \\ 1234567890 & 1234567889 \end{pmatrix}.$$

Here only A_1 produces a smaller coefficients matrix. A_1 it hence the only and best candidate. We check that $A_1^{-1}A_2^{-1}\sigma\mathbf{A} = A_3$. Therefore $\sigma\mathbf{A} = A_2A_1A_3$ and $\sigma = (2, 1, 3)$.

3 Trapdoor construction

Since Decompose works for $\mathbf{A} \subset \mathfrak{D}$ but (conjecturally) fails for elves, it is natural to look for a trapdoor transformation (spell) turning dwarves into elves. Our general strategy is to replace each dwarf $A \in \mathbf{A}$ by an elf of the form XAY, where X and Y are secret. This idea is embodied slightly differently in each of the cryptosystem blueprints.

1. *Trapdoor for the direct construction.* Let $D \in \mathfrak{D}$ and $E \in \mathfrak{E}$ be two secrets. For each $A_i \in \mathbf{A}$, let $\overline{A_i} := EA_iY$, where $Y := DE^{-1}$.
   Indeed, with this choice, we have:

$$\sigma\overline{\mathbf{A}} = \prod_{i=0}^{k-1} \overline{A}_{\sigma(i)} = \prod_{i=0}^{k-1} EA_{\sigma(i)}Y = E\left(\prod_{i=0}^{k-1} A_{\sigma(i)}D\right)E^{-1}.$$

   The trapdoor consists in computing $E^{-1}(\sigma\overline{\mathbf{A}})ED^{-1}$, which is a product of dwarves only — more specifically the k dwarves $A_{\sigma(0)}, \ldots, A_{\sigma(k-1)}$ and $k-1$ copies of the dwarf D[8]. Therefore this product can be Decompose-d.

[8] We do not recover *exactly* $\sigma\mathbf{A}$ through this process, but it is sufficient for correct decryption with well-chosen parameters.

2. *Trapdoor for the alternating construction.* Let $E_0, \ldots, E_k$ be secret distinct elves. For each $A_i^b \in \mathbf{A}^b$ let $\overline{A}_i^b := E_i A_i^b E_{i+1}^{-1}$. With this choice, we have:

$$m\overline{\mathbf{A}} = \prod_{i=0}^{k-1} \overline{A}_i^{m_i} = \prod_{i=0}^{k-1} E_i A_i^{m_i} E_{i+1} = E_0(m\mathbf{A})E_k^{-1}.$$

The trapdoor consists in computing $E_0^{-1}(m\overline{\mathbf{A}})E_k$ which yields $m\mathbf{A}$. The latter is a product of dwarves and can hence be Decompose-d.

Remark 4 (Why D in the direct construction trapdoor?). A natural question is whether we could have taken $D = 1$ (the identity matrix) in the trapdoor, as this would halve the number of products in the ciphertext. Another natural question is whether we need to use the *same* D for all elements of the public key, or we can instead have a unique D_i for each $\overline{A}_i$. These two variations are indeed legitimate and strike different tradeoffs; the former seems to make cryptanalysis simpler, while the latter results in a very large private key while only making cryptanalysis marginally harder.

4 Formal description of the algorithms

We use in this section the following standard notations: assignment (and, if necessary, unpacking) is denoted as $\leftarrow$, checking for equality is denoted by $=$, and uniformly sampling an element from a set is denoted $\xleftarrow{\$}$. We extend the latter notation to tuples, to mean sampling *without replacement*: for instance, $(x, y) \xleftarrow{\$} \mathbf{X}$ does not have the same effect as $x \xleftarrow{\$} \mathbf{X}$ followed by $y \xleftarrow{\$} \mathbf{X}$. The symbol $\perp$ means "failure" and should be treated as an error condition.

Our algorithms are provided public parameters $\mathsf{pp} = (n, p, k, \alpha)$, that have been chosen to meet a certain security level λ (see Section 5 for more about this).

Both variants described in Figures 2 and 3 call Decompose. Note that Decompose can be optimized to fit either the direct or the alternating cryptosystem. Indeed the general algorithm iterates overall all matrices in $\mathbf{A}$. For the direct cryptosystem, we know in advance that every other matrix in the decomposition should be D; for the alternating cryptosystem, there are only two matrices to distinguish between, at any given rank: A_i^0 versus A_i^1. Using these optimized versions of Decompose not only makes it faster, it also substantially reduces η (and thereby, the likelihood and cost of any backtracking). This algorithm is described in Figure 4 and has two new arguments: the current position $0 \leq i \leq k-1$ and a *selection function* Sel taking as input i and the set of matrices $\mathbf{A}$, and returning a subset of $\mathbf{A}$ that can be found at this position:

- For the direct cryptosystem, $\mathsf{Sel}(i, \mathbf{A})$ returns $\mathbf{A}$ if i is even, otherwise $\{D\}$;
- For the alternating cryptosystem, $\mathsf{Sel}(i, \mathbf{A})$ returns $\{A_i^0, A_i^1\}$.

Remark 5. The decomposition algorithm for A can be modified to allow for faster decryption, at the cost of some precomputations. Indeed, at every position i there are only two matrices in $\mathsf{Sel}(i, \mathbf{A})$: should the first one fail, we may *assume* that the other one passes (we don't check it directly). For instance, given $M = A_0^1 A_1^0 A_2^1$, the first iteration computes $(A_0^0)^{-1}M$ and compares it to M — the comparison fails and therefore $m_0 \neq 0$. Rather than confirming that $(A_0^1)^{-1}M \leq M$, we *assume* that it does, and directly pass to test:

$$(A_0^1 A_1^0)^{-1} M \stackrel{?}{\leq} M.$$

If that comparison succeeds, we found $m_0 m_1 = 10$; otherwise we *assume* $m_1 = 1$ and pass to test :

$$(A_0^1 A_1^1 A_2^0)^{-1} M \stackrel{?}{\leq} M$$

and so forth. The matrices $(A_0^1 \dots A_j^1 A_{j+1}^0)^{-1}$ can be precomputed. Note however that this version of $\mathsf{Decompose}$ may never return $\perp$, even when given a product of matrices *not in* $\mathbf{A}$, and therefore does not return a valid decomposition in all cases.

D.KeyGen(pp):	D.Encrypt(pp, pk, σ):	D.Decrypt(pp, sk, C):
1. $(A_0, \dots, A_{k-1}, D) \xleftarrow{\$} \mathfrak{D}$ 2. $E \xleftarrow{\$} \mathfrak{E}$ 3. **for** $i = 0$ **to** $k-1$: $\overline{A}_i \leftarrow E A_i D E^{-1}$ 4. sk $\leftarrow (E, D, \{A_i\})$ 5. pk $\leftarrow \{\overline{A}_i\}$ 6. **return** sk, pk.	1. $\overline{\mathbf{A}} \leftarrow$ pk 2. $C \leftarrow \sigma\overline{\mathbf{A}}$ 3. **return** C.	1. $(E, D, \mathbf{A}) \leftarrow$ sk 2. $T \leftarrow E^{-1} C E D^{-1}$ 3. $\mathbf{A}' \leftarrow \mathbf{A} \cup \{D\}^{k-1}$ 4. $S \leftarrow \mathsf{Decompose}(\mathbf{A}', T)$ 5. **if** $S = \perp$ **return** $\perp$ 6. $(A_{i_0}, \dots, A_{i_{k-1}}) \leftarrow S$ 7. $\sigma \leftarrow (i_0, \dots, i_{k-1})$ 8. **return** σ

Fig. 2. Direct cryptosystem: D.KeyGen, D.Encrypt, and D.Decrypt.

5 Security discussion and parameter selection

Let us turn to the selection of parameters (p, n, k, α) for each cryptosystem, as well as to the resulting key and ciphertext sizes as a function of the security level λ. We give examples for $\lambda = 128$.

5.1 Preliminaries

We need the following results:

A.KeyGen(pp):

1. **for** $i = 0$ **to** $k-1$:
   $(A_i^0, A_i^1) \xleftarrow{\$} \mathfrak{D}$
   with $\det A_i^0 = \det A_i^1$.
2. $(E_0, \ldots, E_k) \xleftarrow{\$} \mathfrak{E}$
3. **for** $i = 0$ **to** $k-1$:
   for $b = 0$ **to** 1:
   $\overline{A}_i^b \leftarrow E_i A_i^b E_{i+1}^{-1}$
4. $\mathsf{sk} \leftarrow (E_0, E_k, \{A_i^b\})$
5. $\mathsf{pk} \leftarrow \{\overline{A}_i^b\}$
6. **return** sk, pk.

A.Encrypt(pp, pk, m):

1. $\overline{\mathbf{A}} \leftarrow \mathsf{pk}$
2. $C \leftarrow m\overline{\mathbf{A}}$
3. **return** C.

A.Decrypt(pp, sk, C):

1. $(E_0, E_k, \mathbf{A}) \leftarrow \mathsf{sk}$
2. $T \leftarrow E_0^{-1} C E_k$
3. $T \leftarrow \mathsf{Decompose}(\mathbf{A}, T)$
4. **if** $T = \perp$ **return** $\perp$
5. $(A_{i_0}^{m_0}, \ldots, A_{i_{k-1}}^{m_{k-1}}) \leftarrow T$
6. $m \leftarrow (m_0, \ldots, m_{k-1})$
7. **return** m.

Fig. 3. Alternating cryptosystem: A.KeyGen, A.Encrypt, and A.Decrypt.

Input: p prime, $n > 1$, M an $n \times n$ matrix over $\mathbb{F}_p$,
$\mathbf{A} = \{A_0, \ldots, A_{k-1}\}$ invertible matrices over $\mathbb{F}_p$
Output: $\{A_{i_0}, \ldots, A_{i_{k-1}}\}$ such that $M = \prod A_{i_j}$, or $\perp$

DecomposeS($\mathbf{A}, M, i$, Sel):

1. **if** $|\mathbf{A}| = 0$ **return** $\perp$.
2. **if** $|\mathbf{A}| = 1$ and $M \notin \mathbf{A}$ **return** $\perp$ **else return** $\mathbf{A}$.
3. **for** $A \in \mathsf{Sel}(i, \mathbf{A})$,
   3.1 $M' \leftarrow A^{-1}M$.
   3.2 **if** $M' \leq M$,
      3.2.1. $\mathbf{L} \leftarrow \mathsf{DecomposeS}(\mathbf{A} - \{A\}, M', i+1, \mathsf{Sel})$.
      3.2.2. **if** $\mathbf{L} = \perp$ **return** $\perp$ **else return** $\{A\} \cup \mathbf{L}$.
4. **return** $\perp$

Fig. 4. Rank-aware decomposition algorithm over $\mathbf{A}$ in $\mathbb{F}_p$.

Lemma 1. *Let $n > 0$, p prime, $0 < x \leq p$. There are:*

$$\beta(n, x) := \prod_{i=0}^{n-1} \left(x^n - x^i\right) - (x - 1)$$

$n \times n$ matrices over $\mathbb{F}_p$ which are invertible, not in the center, and have coefficients at most x (excluded). Consequently, there are $|\mathfrak{E}| = \beta(n, p)$ elves and $|\mathfrak{D}| = \beta(n, \alpha + 1)$ dwarves.

Proof. The proof relies on the classical result that $Z(\mathrm{GL}_n(\mathbb{F}_p)) = \mathbb{F}_p^{\times}\mathbf{1}$ together with an adaptation of the fact that $\mathrm{GL}_n(\mathbb{F}_p) = (p^n - 1)(p^n - p)\cdots(p^n - p^{n-1})$ to basis vectors having bounded entries.

Lemma 2. *Consider the* entry-wise *infinity norm*[9] *on $n \times n$ matrices, defined by $\|M\|_\infty := \max_{i,j} |M_{i,j}|$. Then for any two $n \times n$ matrices M, N we have $\|MN\|_\infty \leq n\|M\|_\infty\|N\|_\infty$.*

Corollary 1. *The product of k dwarves has infinity norm at most $\alpha^k n^{k-1}$.*

Remark 6. Corollary 1 is a conservative estimate — in fact, its upper bound cannot be reached, because we constrain dwarves to be invertible. The only way to attain the stated bound would be with matrices having all entries equal to α — which would be neither invertible, nor distinct.

For instance, with $n = 9$, $\alpha = 1$, $k = 40$, Corollary 1 gives an upper bound of about 2^{124}; but in reality that product doesn't exceed 2^{90}.

5.2 What can be said about security?

While we do not have any formal security reductions for the proposed schemes, we note the following observations as a collection of arguments rather than a thorough structured analysis:

Algebraic relations. We adopt here an algebraic point of view to identify what information is available to an attacker through multiplicative manipulations of the public material, which seems to be the most natural approach against such a highly structured construction.

The direct construction provides many equations for an attacker to play with: in particular, it should in principle be possible to eliminate E by combining the $\overline{A}_i$, resulting in a problem only involving dwarves. We suspect that *an appropriate use of Coppersmith's algorithm (or a variant thereof) might then recover the secret $\mathbf{A}_i^b s$*, if there aren't too many variables. Independently, an attacker successfully recovering a product of the form $A_i A_j^{-1}$ for any two indices i, j[10] can attempt solving the conjugacy search problem (CSP), to obtain E. We *conjecture* that for large enough parameters, these attacks are impractical. Nevertheless, the possibility of their existence is one motivation to also consider the alternating construction.

Indeed the alternating construction leaves much fewer equations for an attacker to hook onto. Nonetheless, attackers still obtain *some* information, as the following lemma shows:

Lemma 3. *From the alternating trapdoor, attackers can obtain the values of all products of the form:*

$$E_0(A_0^1 A_1^1 \cdots A_\ell^1)(A_{\ell+1}^1 (A_{\ell+1}^0)^{-1})(A_0^1 A_1^1 \cdots A_\ell^1)^{-1} E_0^{-1}$$

[9] This is *not* the norm induced on matrices by the infinity norm on vectors, and in particular it is not submultiplicative.

[10] Or equivalently $A_i D^{-1}$ for any index i.

for all $\ell = 0, \ldots, k-2$.

Proof. Strong induction, see Appendix B.

This gives $k-1$ equations, but the number of unknowns increases quadratically; even though the remaining unknowns are *known to be small*. Again, it seems hard to exploit this information algorithmically for cryptanalytic purposes.

Remark 7. One key aspect of the alternating cryptosystem, which remains true before and after using this trapdoor, is that different ciphertexts will have different determinants. This could be a source of leakage, as it presents the attacker with an instance of a multiplicative knapsack problem over $\mathbb{F}_p$. This can be avoided by selecting the dwarves A_i^0 and A_i^1 to have the same determinant, *and* each elf E_i to be of same determinant x_i, at each position i. The exact values of those determinants is unimportant.

Note that determinant values are far from being uniformly distributed. This effect is even more striking with small α values[11]. Consequently, we can easily find matrices with the desired property and avoid this leakage.

It may be that we are too cautious here, and that this leakage is actually unexploitable in practice for large enough parameters.

Determinant ratio attacks. Consider two public-key elements in the alternating cryptosystem $\overline{A}_i^0, \overline{A}_i^1$ at some rank i and let:

$$\Delta = \frac{\det \overline{A}_i^1}{\det \overline{A}_i^0} \bmod p = \frac{\det A_i^1}{\det A_i^0} \bmod p$$

Because decryption must be possible we see that the following holds in $\mathbb{Z}$:

$$\Delta \cdot \det A_i^0 = \det A_i^1$$

It is hence possible to infer the secret-key by solving a Diophantine equation of total degree n in n^2 unknowns. Even though the equation is known to have a small solution, to the best of our knowledge, there is no general method for finding it efficiently.

Source entropy ratio. Collect all the randomness used in generating a key pair and call this the "source entropy". The source entropy will be shared between the private and public key, and some will not feature in either. Let Ξ be the ratio between the information consumed to produce that key and the size of the public key, i.e.:

$$\Xi := \frac{\text{source entropy}}{\text{public key size}}.$$

[11] For $\alpha = 1$ and $n = 9$, the distribution of dwarf determinants has a Shannon entropy of 4 bits, regardless of p.

Although there may be no concrete way to exploit a low value of Ξ nor state that a scheme a high Ξ is "secure", we make the heuristic argument that cryptanalysis is "plausibly harder" when Ξ is larger.

The new cryptosystems have respectively:

$$\Xi_{\mathsf{D}} = \frac{(k+1)\log_2 \beta(n, 1+\alpha) + \log_2 \beta(n,p)}{k \log_2 \beta(n,p)}$$
$$\Xi_{\mathsf{A}} = \frac{2k \log_2 \beta(n, 1+\alpha) + (k+1)\log_2 \beta(n,p)}{2k \log_2 \beta(n,p)}$$

Note that $\Xi_{\mathsf{A}} > \Xi_{\mathsf{D}}$ which seems to indicate that A is a seemingly "stronger" variant as per this criterion. The effect of α is negligible at practical security levels λ.

Normal forms and divisors While we can easily select matrices having the same determinant, there could be additional invariants that our trapdoor doesn't hide, in particular similarity invariants. We describe here two lesser known invariants, which seem to yield some information about the plaintext in theory, although we could not make a concrete attack from them.

Let M be a matrix, its *Smith normal form* (SNF, [Smi62, Div20, Tou13]) is given by three matrices B, T, S such that $BMT = S$ and

$$S = BMT = \mathrm{diag}(\gamma_1, \gamma_2, \ldots, \gamma_r, 0, \ldots, 0)$$

with r is the matrix rank[12]. Applied to invertible matrices, this normal form is diagonal. While the SNF of a matrix product *is not* the product of the factors' SNFs, there are relationships between them [MU72]. Most of these properties are only valid over $\mathbb{Z}$ and break down over finite fields. We could not turn the remaining properties into a concrete attack, but there could be a thread to pull there.

A related potential source of leakage is given by *determinantal divisors*: let M be an $n \times n$ matrix and $i \in \{1, \ldots, n\}$. Choose (in all possible $\binom{n}{i}^2$ ways) i row indices and i column indices, and compute all the determinants of the submatrices constructed from these choices. Take the gcd of all of these determinants, written $d_i(M)$, and called the *i-th determinantal divisor* of M. We have $\gamma_i = d_i(M)/d_{i-1}(M)$ where γ_i are the diagonal entries of M's Smith normal form, and $d_0(M) = 1$.

It is known [New97, p. 371] (see also [New82]) that $d_i(AB) = d_i(A)d_i(B)$ for all $i \in \{1, \ldots, n\}$. The successive gcds quickly sanitize information from high-i determinantal divisors. We thus conjecture that this property is also inexploitable by attackers.

Nonetheless, the observations just made indicate that the algebraic properties underlying the proposed cryptosystems should be carefully investigated.

[12] If computed over $\mathbb{Z}$ rather than over $\mathbb{F}_p$, we also have $\gamma_i \mid \gamma_{i+1}$ for $1 \leq i < r$.

5.3 Parameters for D

The parameters (p, n, k, α) must satisfy several constraints:

- Brute-force exhaustion of the ciphertext should be intractable: $k! \geq 2^\lambda$.
- Encryption must be injective, which implies that $|\mathfrak{E}| > k!$.
- There should be enough dwarves to choose from, both to avoid attacks that guess a dwarf and to have enough choice in the private key: $|\mathfrak{D}| > 2^\lambda$.
- Decomposition works under the assumption that $\|E^{-1}(\sigma\overline{\mathbf{A}})DE^{-1}\|_\infty < p$, which is guaranteed by Corollary 1 if $\alpha^{2k}n^{2k-1} < p$[13,14].

It is challenging to find parameters simultaneously satisfying all these constraints. Identifying the most relevant trade-offs is an interesting question in its own right. As highlighted in Appendix C, there are some forbidden parameter ranges.

Here is one strategy that seems to work: start from $\beta(n, \alpha+1) > 2^\lambda$ which relates n and α, then select the values such that $n\alpha$ is smallest; then fix k and select the smallest prime p bigger than $\alpha^{2k}n^{2k-1}$. Then, by design, all the above constraints are satisfied[15].

The secret key consists in $k+1$ dwarves and one elf, the public key consists of k elves, the ciphertext is an elf.

5.4 Parameters for A

The constraints are slightly different:

- Brute-force exhaustion of the ciphertext should be intractable: $k \geq \lambda$.
- Encryption must be injective, which implies that $|\mathfrak{E}| > 2^k$.
- There should be enough dwarves to choose from, both to avoid attacks that guess a dwarf and to have enough choice in the private key: $|\mathfrak{D}| > 2^\lambda$.
- Decompose works under the assumption that $\|E_0^{-1}(m\overline{\mathbf{A}})E_k\|_\infty < p$, which is guaranteed by Corollary 1 if $\alpha^k n^{k-1} < p$[16].

Finding n and α can be done in the same way as for the direct cryptosystem. Once k is fixed we can select the smallest prime p bigger than $\alpha^k n^{k-1}$. Then, by construction, all the above constraints are satisfied.

The secret key consists in $2k$ dwarves and two elves, the public key consists in $2k$ elves, the ciphertext is an elf.

[13] Note that neither of these conditions is necessary, in light of Remark 6, although we treat them as such.

[14] This condition must be adapted in case we select the $D = 1$ variant discussed in Remark 4

[15] Additional parameter sets, complete with worked out examples, can be found in Appendix D.

[16] Here again, Remark 6 indicates that this choice is overly conservative.

5.5 Proposed parameter sets

We provide in tables 1 and 2 four sets of parameters for each construction, dubbed "toy", "challenge", "recommended", "large". These parameters were chosen minimally for a given target security parameter according to the procedure described above. They are intended as targets for cryptanalysis, with increasing levels of (conjectured) difficulty.

We provide private key (sk), public key (pk), and ciphertext (C) sizes in bytes.

Table 1. Suggested parameter sets for D.

Version	Toy	Challenge	Recommended	Large
λ	16	64	128	512
k	9	21	35	99
n	4	8	10	24
α	2	2	2	2
p	$2^{53}+5$	$2^{167}+83$	$2^{302}+307$	$2^{1105}-1335$
pk (B)	7776	28 224	132 580	7 876 440
sk (B)	904	1696	4688	93 960
C (B)	864	1344	3788	79 560

Table 2. Suggested parameter sets for A.

Version	Toy	Challenge	Recommended	Large
λ	16	64	128	512
k	16	64	128	512
n	4	8	10	24
α	2	2	2	2
p	$2^{47}+5$	$2^{255}-19$	$2^{553}+549$	$2^{2859}+641$
pk (B)	3072	261 120	1 772 800	210 862 080
sk (B)	320	6128	20 250	559 296
C (B)	96	2040	6925	205 920

6 Open questions & further research

The novelty of the proposed constructions naturally raises a wealth of open questions. We hereafter list a few:

Security reduction. A very natural and somewhat frustrating question is to determine whether our constructions' security can be related to the hardness of existing, well-studied problems. Indeed it is possible that we overlooked a very natural, simple and devastating attack: a security proof would alleviate such concerns.

Security against active attacks. Being deterministic, our constructions can't hope to achieve semantic security, let alone stronger properties such as IND-CCA2. Furthermore, there are homomorphisms (see below) that may allow for some ciphertext malleability. This is not in itself a death blow, as the same can be said of e.g., textbook RSA, and there are known ways around this limitation[17,18,19].

Public-key compression. Shortening pk is beneficial for two reasons. The first is the quadratic size of each A_i. In addition, publishing less A_is *may* expose less information to the attacker (if the specific way of using a lesser number of public-key elements does not introduce new risks!).

We suggest the following idea applicable to the alternating version. Instead of encoding only one bit using either A_i^0 or A_i^1, pick for each position i a family of u pairs of scalars:

$$(\tau_{v_{i,0}}, \rho_{v_{i,0}}), (\tau_{v_{i,1}}, \rho_{v_{i,1}}), \ldots, (\tau_{v_{i,u-1}}, \rho_{v_{i,u-1}}) \text{ such that } \forall j,\ \gcd(\tau_{v_{i,j}}, \rho_{v_{i,j}}) = 1$$

Use a different linear combination $\tau_{v_{i,u-1}} A_i^0 + \rho_{v_{i,u-1}} A_i^0$ to encode each plaintext chunk. Typically, if we allow each chunk to be represented by a couple $(\tau_{v_{i,j}}, \rho_{v_{i,j}})$ such that $\max(\tau_{v_{i,j}}, \rho_{v_{i,j}}) \leq 41$ we can encode at each position $1061 \simeq 2^{10}$ values, i.e. 10 bits. This shortens the public-key by a factor of 10 but increases decryption time by a factor of 1024. cf. Table 3. Note that the slowdown can be reduced a factor of n^2 by just testing, during $\mathsf{Decompose}$, one coordinate of the intermediate matrix and if a decrease is witnessed, confirm the decrease over all other coordinates.

We did not investigate the security of this variant nor its impact on p.

Key generation. A central assumption underlying our schemes is that private keys allowing decryption can be efficiently generated. The exact requirements are

[17] Our cryptosystems do not provide a straightforward pseudo-random permutation, so techniques such as OAEP or OAEP+ are not directly applicable [Sho02].

[18] There seems to have been multiple attempts to extend OAEP to work for generic hard to invert functions (e.g., [Jon02, KI02]), but to the best of our knowledge they have not made their ways into peer-reviewed journals or conferences. We recall in Appendix A one such mechanism from [FN19].

[19] Neither OAEP nor its variants are quantum secure, unless appropriate modifications are applied, cf. [Ebr22].

ϕ	1	2	3	5	7	10	14	20	29	41
η	1	2	3	4	5	6	7	8	9	10
ζ	1	4	8	16	32	64	128	256	512	1024

Table 3. $\phi = \max(\tau_{v_{i,j}}, \rho_{v_{i,j}})$. η is the factor by which pk size is reduced. The decryption slowdown is $\zeta = 2^\eta$.

not the same for the direct and the alternating cryptosystem. Empirical evidence suggests that for suggested parameters, random invertible matrices (obtained e.g. by reversing LU decomposition) almost always work. However, it is also easy to fix p, n, and α small enough to make finding a valid key impossible (see Appendix C). Therefore the following questions are open: under what *exact* conditions on $\mathbf{A}$ is $\sigma \mapsto \sigma\mathbf{A}$ (resp. $m \mapsto m\mathbf{A}$) injective? Can we always find such matrices efficiently? Can we force them to have a certain determinant?

Finding alternative key generation methods is also an interesting research direction. The following example proceeds in several steps, assuming that $k-1$ is odd (i.e., that pk contains an even number of elements).

Example 2. Consider $A_4^0, A_4^1, A_5^0, A_5^1 \in \mathfrak{D}$, used to encode two successive plaintext bits (say bits 4 and 5) as in the alternating construction. We wish all four products $A_4^i A_5^j$ for $(i,j) \in ((0,0),(0,1),(1,0),(1,1))$ to be simultaneously dwarves.

Pick four $M_{4,5}^{i,j} \in \mathfrak{D}$ and set the following system of equations:

$$A_4^i A_5^j = M_{4,5}^{i,j} \bmod p.$$

Here, there are $4n^2$ unknowns in $4n^2$ equations; these unknowns are the coefficients of A_4^0, A_4^1, A_5^0 and A_5^1. We can solve it exactly, and having solved those equations we repeat the process for matrices at other ranks. Upon encryption we now have:

$$\begin{aligned} m\overline{\mathbf{A}} &= E_0(A_0^{m_0}A_1^{m_1})\cdots(A_{k-2}^{m_{k-2}}A_{k-1}^{m_{k-1}})E_k^{-1} \\ &= E_0 M_{0,1}^{m_0,m_1}\cdots M_{k-2,k-1}^{m_{k-2},m_{k-1}}E_k^{-1} \\ &= E_0(m\overline{\mathbf{A}})E_k^{-1} \end{aligned}$$

where each matrix $M_{i,j}^{a,b} \in \mathfrak{D}$ by design. This could result in a more compact ciphertext, or improve decryption, as the total number of products decreased from $k-1$ down to $k/2-1$. Whether this can be fully fleshed out securely is left as an open question.

Homomorphisms. The schemes are not straightforwardly multiplicatively homomorphic; for instance $\sigma_1\mathbf{A}\sigma_2\mathbf{A} \neq (\sigma_1\sigma_2)\mathbf{A}$ in general. Yet there is a natural map sending (σ_1, σ_2) to a permutation τ over $\mathbf{A}^2 := \mathbf{A} \sqcup \mathbf{A}$ such that $\sigma_1\mathbf{A}\sigma_2\mathbf{A} = \tau\mathbf{A}^2$. The consequences of this homomorphism — and others — have yet to be investigated.

Digital signatures. An interesting challenge consists in deriving digital signatures based on the proposed trapdoor. Because the proposed trapdoor is not a permutation, signing by message decryption is impossible. Similarly, the deterministic nature of the proposed cryptosystems seems to ban the use of Fiat–Shamir transforms. Signatures can also be obtained using ad hoc constructions (e.g. [BFJN20]) but we could not adapt any such approaches so far.

KEMs. We can build key-encapsulation mechanisms from our constructions, for instance as follows:

Starting from the alternating cryptosystem[20], select a prime $t < \alpha$, and publish as part of pk the matrices $V_i^b := S_i A_i^b S_{i+1} \bmod t$ where the S_i are new random elves. Let $\mathbf{V}$ denote the set of the V_i^b. Encapsulation is performed by computing $C = m\overline{\mathbf{A}}$ for a randomly chosen m to get the ciphertext, and $K = \mathcal{H}(m\mathbf{V} \bmod t)$ to get the shared key, where $\mathcal{H}$ is a hash function. Decapsulation[21] computes:

$$K = \mathcal{H}\left(S_0 \left(E_0^{-1} C E_k \bmod p\right) S_k \bmod t\right).$$

Other rings. We may wonder whether it is necessary to use matrices at all: indeed, nothing in our alternating construction in particular relies on matrix-specific properties. One tempting idea is for instance to use a ring of polynomials instead, which would reduce the number of entries and the cost of arithmetic operations. However our attempts to do so have all resulted in reduced η or insecure variants. Of course, further rings such as quaternions or even octonions could be considered, and there could be room for new discoveries in such generalizations. Working modulo a composite p (typically an RSA modulus) might also give birth to new variants.

Higher dimensions. The scheme can be generalized to higher dimension constructs such as tensors – thereby yielding cubic monomials and hence higher degree equations. Because many unknowns in our cryptosystem are small with respect to p, increasing the degree of the equations is likely to better resist resolution attempts using Gröbner bases, linearization attacks and Coppersmith's algorithm [Cop96b,Cop96a,Cor04,BJ07,Cor07]. Note that this comes at a significant increase of pk which is now $O(kn^d \log p)$ space. We implemented the proposed algorithms on 3D tensors to assess soundness (the possibility to successfully encrypt and decrypt) but did not further investigate the security of such generalizations.

Repeated indices and matrix powers. We made the simplifying hypothesis that all matrices were distinct. Allowing for repeated matrices (at different, nonadjacent positions), or even matrix powers could increase the schemes' bandwidth, or

[20] The same idea can be adapted to the direct cryptosystem.

[21] Note that this variant does not use $\mathsf{Decompose}$ and that decapsulation is very fast and that the decapsulation key footprint is very low.

reduce keysize. The entropy loss caused by such variants might affect both security and η and should hence be carefully assessed. In a sense, this is reminiscent of the Tillich–Zémor construction and may suffer from similar attacks.

Compression. The private key can be generated on the fly from a random seed, and only the inverse matrices need to be stored in full for decryption or calculated on the fly and dropped after use.

Inserting full-size elements into the A_i. An interesting question is that of the *survival* of dwarfism in the product when full-size elements are inserted into some dwarves. If the first dwarf has full size elements in all lines except one, then matrix multiplication by standard dwarves preserves this property and still allows decryption by just looking at the surviving line. Alas, this observation cannot be exploited to inject extra entropy because the attacker can strip-off the first matrix and face the problem of decrypting a cryptosystem encrypting $k-1$ bits. Any strategies allowing to overcome this problem are interesting research directions.

References

AW20. Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication, 2020. 2

BFJN20. Éric Brier, Houda Ferradi, Marc Joye, and David Naccache. New number-theoretic cryptographic primitives. *Journal of Mathematical Cryptology*, 14(1):224–235, 2020. 17

BG08. Jean Bourgain and Alex Gamburd. Uniform expansion bounds for cayley graphs of. *Annals of Mathematics*, pages 625–642, 2008. 3

BJ07. Aurélie Bauer and Antoine Joux. Toward a rigorous variation of Coppersmith's algorithm on three variables. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007*, pages 361–378, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. 17

BKT04. Jean Bourgain, Nets Katz, and Terence Tao. A sum-product estimate in finite fields, and applications. *Geometric & Functional Analysis GAFA*, 14(1):27–57, 2004. 3

Bro21. Daniel R.L. Brown. Plactic signatures (insecure?). Cryptology ePrint Archive, Paper 2021/938, 2021. https://eprint.iacr.org/2021/938. 1

BS92. László Babai and Ákos Seress. On the diameter of permutation groups. *European journal of combinatorics*, 13(4):231–243, 1992. 3

Cop96a. Don Coppersmith. Finding a small root of a bivariate integer equation: Factoring with high bits known. In *Advances in Cryptology - EUROCRYPT '96*, pages 178–189. Springer Berlin Heidelberg, 1996. 17

Cop96b. Don Coppersmith. Finding a small root of a univariate modular equation. In *Advances in Cryptology - EUROCRYPT '96*, pages 155–165. Springer Berlin Heidelberg, 1996. 17

Cor04. Jean-Sébastien Coron. Finding small roots of bivariate integer polynomial equations revisited. In *Advances in Cryptology - EUROCRYPT 2004*, pages 492–505. Springer Berlin Heidelberg, 2004. 17

Cor07. Jean-Sébastien Coron. Finding small roots of bivariate integer polynomial equations: A direct approach. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 379–394, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. 17

Div20. Jose Divasón. A verified algorithm for computing the smith normal form of a matrix. *Arch. Formal Proofs*, 2020, 2020. 12

Ebr22. Ehsan Ebrahimi. Post-quantum security of plain OAEP transform. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part I*, volume 13177 of *Lecture Notes in Computer Science*, pages 34–51. Springer, 2022. 15

FN19. Houda Ferradi and David Naccache. Integer reconstruction public-key encryption. In Yi Mu, Robert H. Deng, and Xinyi Huang, editors, *Cryptology and Network Security*, pages 412–433, Cham, 2019. Springer International Publishing. 15, 20

HS03. Dennis Hofheinz and Rainer Steinwandt. A practical attack on some braid group based cryptographic primitives. In Yvo Desmedt, editor, *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, volume 2567 of *Lecture Notes in Computer Science*, pages 187–198. Springer, 2003. 2

Jon02. Jakob Jonsson. An OAEP variant with a tight security proof. *IACR Cryptol. ePrint Arch.*, page 34, 2002. 15

KI02. Kazukuni Kobara and Hideki Imai. OAEP++ : A very simple way to apply OAEP to deterministic OW-CPA primitives. *IACR Cryptol. ePrint Arch.*, page 130, 2002. 15

Mon22. Chris Monico. Division in the plactic monoid. Cryptology ePrint Archive, Paper 2022/1684, 2022. https://eprint.iacr.org/2022/1684. 1

MU72. Marvin Marcus and Ernest E. Underwood. A Note on the Multiplicative Property of the Smith Normal Form. *J. of research of the Notional Bureau of Standards - B. Mathematical Sciences*, 76B(3 and 4):205–206, 1972. 12

New82. Morris Newman. A result about determinantal divisors. *Linear and Multilinear Algebra*, 11(4):363–366, 1982. 12

New97. Morris Newman. The Smith normal form. *Linear Algebra and its Applications*, 254(1):367–381, 1997. Proceeding of the Fifth Conference of the International Linear Algebra Society. 12

Pei15. Chris Peikert. A decade of lattice cryptography. Cryptology ePrint Archive, Paper 2015/939, 2015. https://eprint.iacr.org/2015/939. 1

PLZG23. Jacques Peyriere, Fengxia Liu, Zhiyong Zheng, and Zixian Gong. Public key cryptosystems based on iterated functions systems, 2023. 2

PQ11. Christophe Petit and Jean-Jacques Quisquater. Rubik's for cryptographers. *IACR Cryptol. ePrint Arch.*, page 638, 2011. 2

Sho02. Victor Shoup. OAEP reconsidered. *J. Cryptol.*, 15(4):223–249, 2002. 15

SHS21. Philipp Stratil, Shingo Hasegawa, and Hiroki Shizuya. Supersingular isogeny-based cryptography: A survey. *Interdisciplinary Information Sciences*, 27(1):1–23, 2021. 1

Smi62. Henry John Stephen Smith. I. On systems of linear indeterminate equations and congruences. *Proceedings of the Royal Society of London*, 1(11):86–89, 1862. 12

Tol12. J.R.R. Tolkien. *The Silmarillion.* HarperCollins, 2012. 4
Tou13. Vasilios Evangelos Tourloupis. Hermite normal forms and its cryptographic applications. Master's thesis, University of Wollongong, 2013. 12

A Generic transformation

The cryptosystems presented in this article are entirely deterministic and therefore cannot hope to be semantically secure. We propose the following hybrid construction to address this point: the message m is encrypted using a symmetric algorithm (e.g., AES) denoted $\mathcal{F}$, with a key derived by hashing m with a random r. Importantly this derivation happens in two steps, and the intermediary value is encrypted using the new public key cryptosystem. Derivation makes use of two independent hash functions $\mathcal{H}, \mathcal{H}'$, which can be implemented concretely using e.g. SHA3 with domain separation.

For decryption, the intermediary value is obtained using the new decryption algorithm, and reveals the secret key used for the symmetric algorithm, so that m can be recovered. At the end of this process, we have also recovered the randomness r, and we can check that the message derives the correct intermediary value. This is summarized in Figure 5.

This construction, inspired by [FN19], is generic, and does not depend on the specific public-key cryptosystem used.

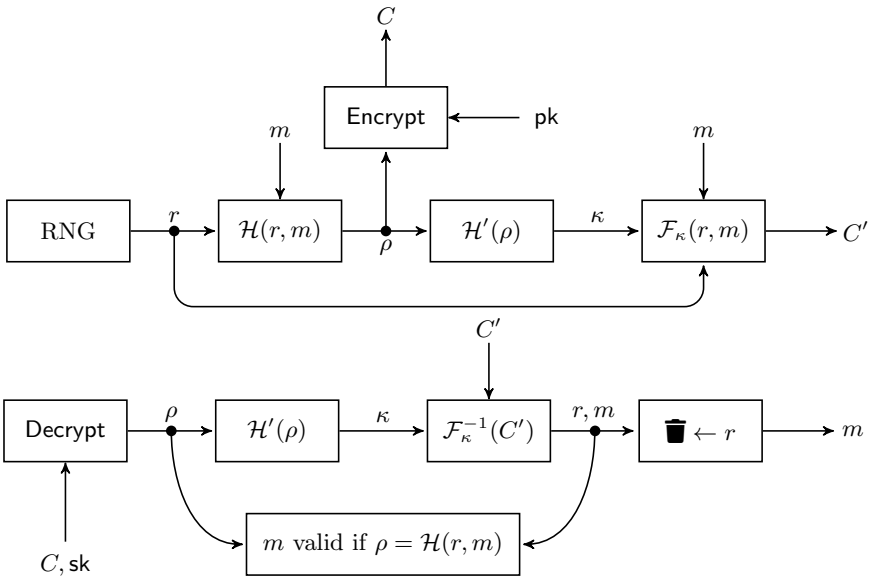

Fig. 5. Schematic view of the proposed generic transformation.

B Proof of Lemma 3

It is convenient, for the context of this proof, to introduce the following shorthand notation:

$$[\![a_0, a_1, \ldots, a_\ell]\!] := A_0^{a_0} A_1^{a_1} \cdots A_\ell^{a_\ell}.$$

Let's first establish the base case: the value $E_0\,[\![1]\!]\,(E_0\,[\![0]\!])^{-1}$ is public. Indeed, this is exactly $\overline{A_0^1}(\overline{A_0^1})^{-1}$. Now assume that the lemma holds for all ranks $\ell = 0, \ldots, r$, and denote by $L(\ell)$ the value that the lemma asserts can be computed. We show that is it established for $r+1$, i.e., that we can express $L(r+1)$ only from previous values of L and public key elements.

By combining the public key elements $\overline{A}_i^b$ for $i = 0, \ldots, r+1$, we always get (up to inversion of the whole expression) a quantity of the form:

$$X_0\,[\![a_0, \ldots, a_r]\!]\,A_{r+1}^0 (E_0\,[\![b_0, \ldots, b_r]\!]\,A_{r+1}^1)^{-1}$$

If $a_0 = 0$, then multiplying the above expression by $L(0)$ ensures that $a_0 = 1$. After that, if $a_1 = 0$ then multiplying the expression by $L(1)$ ensures that $a_1 = 1$. We can continue this process until $a_0 = a_1 = \cdots = a_r = 1$. Multiplying on the right by appropriate inverses of $L(0), \ldots, L(r)$ similarly brings $b_0 = \cdots = b_r = 1$.

Consequently, we have obtained $L(r+1) = E_0\,[\![1, \ldots, 1]\!]\,A_{r+1}^1 (E_0\,[\![1, \ldots, 1]\!]\,A_{r+1}^0)^{-1}$, which concludes the proof by strong induction.

Remark 8. This shows that if we combine in *any multiplicative way* further information coming of any additional public-key element, we introduce two unknown matrices to the system while adding one equation only.

Evidently there are two caveats here:

1. We consider only key products and not any operator (e.g. matrix addition) in general.
2. While we add at each step $2n^2$ unknowns, those unknowns are known to be small.

A complete example is worked out in Appendix D.3.

C The problem with small matrices over small fields

When $n = p = 2$ the only invertible matrices are:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad M_1 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad M_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

$$J = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad M_3 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \qquad M_3^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

Note that this fixes $\alpha = 1$. In particular:

$$(M_1 M_2)^{-1} = (JM_1)^{-1} = M_2 M_1 = JM_2 = M_3$$
$$M_3 M_3^{-1} = M_1^2 = M_2^2 = J^2 = M_3^3 = I$$

Therefore if M_1 and M_2 belong to the the private key, then J, M_3, and M_3^{-1} cannot as this would break the injectivity of encryption. In fact, selecting any two matrices in the list excludes the others (of course I can never be part of this key). But then, it is impossible to generate a public key! Indeed, doing so would introduce more invertible matrices into the product, which inevitably causes collisions.

Evidently, the parameters $p = n = 2$ are anyhow out of the security range but the question of determining how this phenomenon scales-up to higher p, n values is yet to be fully investigated.

D Toy examples

D.1 Direct cryptosystem

The following example artificially use artificially small (insecure) parameters. The point is only to illustrate key generation, encryption, and decryption. Let $\{n, k, \alpha, p\} = \{2, 3, 3, 877\}$.

Key generation. We obtain the following private key:

$$E = \begin{pmatrix} 169 & 315 \\ 450 & 395 \end{pmatrix}, \quad D = \begin{pmatrix} 2 & 3 \\ 3 & 3 \end{pmatrix}, \quad A_0 = \begin{pmatrix} 1 & 3 \\ 2 & 0 \end{pmatrix}, \quad A_1 = \begin{pmatrix} 1 & 2 \\ 3 & 2 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 3 & 2 \\ 1 & 0 \end{pmatrix}.$$

The public key is:

$$\overline{A}_0 = \begin{pmatrix} 235 & 402 \\ 479 & 659 \end{pmatrix}, \qquad \overline{A}_1 = \begin{pmatrix} 416 & 510 \\ 608 & 484 \end{pmatrix}, \qquad \overline{A}_2 = \begin{pmatrix} 36 & 846 \\ 845 & 856 \end{pmatrix}.$$

Encryption. Consider the plaintext $\sigma = (213)$, then the ciphertext is

$$C = \sigma\overline{\mathbf{A}} = \overline{A}_1 \overline{A}_0 \overline{A}_2 = \begin{pmatrix} 521 & 99 \\ 347 & 464 \end{pmatrix}.$$

Decryption. First we compute the temporary value

$$T = E^{-1} C E D^{-1} = \begin{pmatrix} 522 & 248 \\ 810 & 384 \end{pmatrix}$$

then we run the optimized decomposition algorithm of Figure 4 on T. Below we show what happens at each step.

1. ($i = 0$) We test up to three matrices A_0, A_1, A_2:

$$T_0 = A_0^{-1}T = \begin{pmatrix} 405 & 192 \\ 39 & 311 \end{pmatrix}, \qquad T_1 = A_1^{-1}T = \begin{pmatrix} 144 & 68 \\ 189 & 90 \end{pmatrix},$$

$$T_2 = A_2^{-1}T = \begin{pmatrix} 810 & 384 \\ 800 & 425 \end{pmatrix}$$

   We can check that $T_0, T_1 \leq T$. It is true that $T_1 \leq T_0$, and we could use this to select T_1 directly, but for the sake of illustration we will perform backtracking and fork two branches.
2. ($i = 1$, branch 0) We update T with the only available matrix at this step, D:

$$T \leftarrow D^{-1}T_0 = \begin{pmatrix} 511 & 119 \\ 379 & 277 \end{pmatrix}$$

   We see here that this operation has caused an *increase* in coefficients, and therefore this branch is invalid: we prune it.
3. ($i = 1$, branch 1) We update T with the only available matrix at this step, D:

$$T \leftarrow D^{-1}T_1 = \begin{pmatrix} 45 & 22 \\ 18 & 8 \end{pmatrix}$$

4. ($i = 2$) We test up to two matrices A_0, A_2:

$$T_0 = A_0^{-1}T = \begin{pmatrix} 9 & 4 \\ 12 & 6 \end{pmatrix}, \qquad T_2 = A_2^{-1}T = \begin{pmatrix} 18 & 8 \\ 434 & 876 \end{pmatrix}$$

   There is only one admissible choice which is T_0.
5. ($i = 3$) We update T with the only available matrix at this step, D:

$$T \leftarrow D^{-1}T_1 = \begin{pmatrix} 3 & 2 \\ 1 & 0 \end{pmatrix}$$

6. ($i = 4$) There is only one matrix remaining in our list, A_2, and we check that indeed $T = A_2$. The algorithm successfully terminates and has correctly recovered our plaintext.

D.2 Alternating cryptosystem

Set $\{n, k, \alpha, p\} = \{2, 4, 100, 800000011\}$. Pick:

$$A_0^0 = \begin{pmatrix} 24 & 94 \\ 91 & 45 \end{pmatrix} \quad A_1^0 = \begin{pmatrix} 63 & 51 \\ 26 & 57 \end{pmatrix} \quad A_2^0 = \begin{pmatrix} 85 & 42 \\ 16 & 15 \end{pmatrix} \quad A_3^0 = \begin{pmatrix} 44 & 29 \\ 49 & 12 \end{pmatrix}$$

$$A_0^1 = \begin{pmatrix} 27 & 23 \\ 41 & 22 \end{pmatrix} \quad A_1^1 = \begin{pmatrix} 41 & 36 \\ 70 & 38 \end{pmatrix} \quad A_2^1 = \begin{pmatrix} 73 & 69 \\ 87 & 47 \end{pmatrix} \quad A_3^1 = \begin{pmatrix} 52 & 33 \\ 80 & 67 \end{pmatrix}$$

$$E_0 = \begin{pmatrix} 181465853 & 489998889 \\ 586318319 & 354105151 \end{pmatrix} \qquad E_1 = \begin{pmatrix} 737027227 & 204799375 \\ 240734459 & 201952588 \end{pmatrix}$$
$$E_2 = \begin{pmatrix} 462605764 & 533199356 \\ 028731110 & 540728462 \end{pmatrix} \qquad E_3 = \begin{pmatrix} 153582549 & 270321504 \\ 280451465 & 475609880 \end{pmatrix}$$
$$E_4 = \begin{pmatrix} 493998496 & 252943122 \\ 113547886 & 273933344 \end{pmatrix}$$

The public key is therefore:

$$\overline{A}_0^0 = \begin{pmatrix} 430791889 & 308371533 \\ 407818308 & 626147730 \end{pmatrix} \qquad \overline{A}_0^1 = \begin{pmatrix} 167388491 & 602359514 \\ 514933164 & 574622894 \end{pmatrix}$$
$$\overline{A}_1^0 = \begin{pmatrix} 604792527 & 259342305 \\ 521877729 & 466004522 \end{pmatrix} \qquad \overline{A}_1^1 = \begin{pmatrix} 140711117 & 204642304 \\ 589035509 & 414102339 \end{pmatrix}$$
$$\overline{A}_2^0 = \begin{pmatrix} 641418896 & 051951188 \\ 188365934 & 615393743 \end{pmatrix} \qquad \overline{A}_2^1 = \begin{pmatrix} 037157200 & 763989885 \\ 222238244 & 774810537 \end{pmatrix}$$
$$\overline{A}_3^0 = \begin{pmatrix} 535108422 & 653847419 \\ 486148410 & 280095663 \end{pmatrix} \qquad \overline{A}_3^1 = \begin{pmatrix} 119122866 & 072507688 \\ 076310737 & 183717366 \end{pmatrix}$$

Consider the plaintext $m = (1, 0, 1, 0)$. The corresponding ciphertext is:

$$C = \overline{A}_0^1 \overline{A}_1^0 \overline{A}_2^1 \overline{A}_3^0 = \begin{pmatrix} 736131904 & 041083532 \\ 281879757 & 104491974 \end{pmatrix}$$

To decrypt C we first strip-off E_0 and E_4:

$$T_3 = E_0^{-1} C E_4 = \begin{pmatrix} 31637435 & 15068411 \\ 41309110 & 19617490 \end{pmatrix}$$

We try to strip-off A_3^0 or A_3^1:

$$T_3(A_3^0)^{-1} = \begin{pmatrix} 401683\ 284967 \\ 521330\ 374910 \end{pmatrix} \leq T_3(A_3^1)^{-1} = \begin{pmatrix} 748950526\ 383577638 \\ 363505113\ \ 24238044 \end{pmatrix}$$
$$\Downarrow$$
$$T_2 = T_3(A_3^0)^{-1}$$
$$\Downarrow$$
$$T_2(A_2^0)^{-1} = \begin{pmatrix} 744281048\ 796032103 \\ 553236859\ 210961813 \end{pmatrix} \geq T_2(A_2^1)^{-1} = \begin{pmatrix} 2299\ 2688 \\ 3155\ 3345 \end{pmatrix}$$
$$\Downarrow$$
$$T_1 = T_2(A_2^1)^{-1}$$
$$\Downarrow$$
$$T_1(A_1^0)^{-1} = \begin{pmatrix} 27\ 23 \\ 41\ 22 \end{pmatrix} \geq T_1(A_1^1)^{-1} = \begin{pmatrix} 761746477\ 330977107 \\ 390852515\ 113929291 \end{pmatrix}$$
$$\Downarrow$$
$$T_0 = T_1(A_1^0)^{-1} = A_0^1 \Rightarrow m = (1, 0, 1, 0)$$

We instrument this example to also illustrate the leakage through the determinant addressed in the paper. We purposely took in the example dwarves A_i^0, A_i^1 having nonidentical determinants. Compute for instance:

$$\Delta = \frac{\det \overline{A}_0^1}{\det \overline{A}_0^0} \bmod p = \frac{\det A_0^1}{\det A_0^0} \bmod p = 213112125$$

We note that $\det A_0^0 = -349$ and $\det A_0^1 = -7474$ and that $349 \times 7474 = 2608426 < p$. Hence LLL can be used to write ρ as a modular ratio and reveal the determinants of the A_0^b. There are $103020000 \simeq 2^{26.61}$ invertible matrices with coefficients bounded by 100. All matrix pairs whose $\Delta = 213112125$ have determinants equal to $\pm\{349\Delta, 7474\} \bmod p$. Each of the two pairs appears $9702 \simeq 2^{13.24}$ times. Hence, the determinant leaked about 13 entropy bits about the concerned pair of key elements.

D.3 Application of Lemma 3

For the sake of concision, we denote in this example $M^{-1} = \widehat{M}$. We start with one public-key element e.g.:

$$\{\overline{A}_3^0, \overline{A}_3^1\} = \{E_3 A_3^0 \widehat{E}_4, E_3 A_3^1 \widehat{E}_4\}$$

Form $\overline{A}_3^0 \widehat{\overline{A}}_3^1 = E_3 A_3^0 \widehat{E}_4 E_4 \widehat{A}_3^1 \widehat{E}_3 = E_3 A_3^0 \widehat{A}_3^1 \widehat{E}_3$: we are left with n^2 equations in n^2 large unknowns (the elements of E_3) and $2n^2$ small unknowns (the elements of A_3^0 and A_3^1).

Take two successive public-key elements for instance at rank 3 and 4 and form the 4 quantities.$S_{b_3,b_4} = \overline{A}_3^{b_3}\overline{A}_4^{b_4} = E_3 A_3^{b_3}\widehat{E}_4 E_4 A_4^{b_4}\widehat{E}_5 = E_3 A_3^{b_3} A_4^{b_4}\widehat{E}_5$.

At a first resolution step we eliminate E_5. There are 16 ways to do so (of which 4 are trivial that we remove):

$$\begin{array}{lll} S_{0,0}\widehat{S}_{0,1} & S_{0,0}\widehat{S}_{1,0} & S_{0,0}\widehat{S}_{1,1} \\ S_{0,1}\widehat{S}_{0,0} & S_{0,1}\widehat{S}_{1,0} & S_{0,1}\widehat{S}_{1,1} \\ S_{1,0}\widehat{S}_{0,0} & S_{1,0}\widehat{S}_{0,1} & S_{1,0}\widehat{S}_{1,1} \\ S_{1,1}\widehat{S}_{0,0} & S_{1,1}\widehat{S}_{0,1} & S_{1,1}\widehat{S}_{1,0} \end{array}$$

The above set contains 10 distinct matrices of which only 6 are not inverses of others:

$$\begin{cases} E_3 A_3^1 \widehat{A}_3^0 \widehat{E}_3 & (1) \\ E_3 A_3^0 A_4^1 \widehat{A}_4^0 \widehat{A}_3^0 \widehat{E}_3 & (2) \\ E_3 A_3^1 A_4^0 \widehat{A}_4^1 \widehat{A}_3^0 \widehat{E}_3 & (3) \\ E_3 A_3^1 A_4^0 \widehat{A}_4^1 \widehat{A}_3^1 \widehat{E}_3 & (4) \\ E_3 A_3^1 A_4^1 \widehat{A}_4^0 \widehat{A}_3^0 \widehat{E}_3 & (5) \\ E_3 A_3^1 A_4^1 \widehat{A}_4^0 \widehat{A}_3^1 \widehat{E}_3 & (6) \end{cases}$$

Multiplying (2),(3) and (5) by $\widehat{(1)}$ we get:

$$\begin{cases} E_3 A_3^1 \widehat{A}_3^0 \widehat{E}_3 & (7) \\ E_3 A_3^0 A_4^1 \widehat{A}_4^0 \widehat{A}_3^1 \widehat{E}_3 & (8) \\ E_3 A_3^1 A_4^0 \widehat{A}_4^1 \widehat{A}_3^1 \widehat{E}_3 & (9) \\ E_3 A_3^1 A_4^0 \widehat{A}_4^1 \widehat{A}_3^1 \widehat{E}_3 & (10) \\ E_3 A_3^1 A_4^1 \widehat{A}_4^0 \widehat{A}_3^1 \widehat{E}_3 & (11) \\ E_3 A_3^1 A_4^1 \widehat{A}_4^0 \widehat{A}_3^1 \widehat{E}_3 & (12) \end{cases}$$

Eliminating duplicates we get:

$$\begin{cases} E_3 A_3^1 \widehat{A}_3^0 \widehat{E}_3 & (13) \\ E_3 A_3^0 A_4^1 \widehat{A}_4^0 \widehat{A}_3^1 \widehat{E}_3 & (14) \\ E_3 A_3^1 A_4^0 \widehat{A}_4^1 \widehat{A}_3^1 \widehat{E}_3 & (15) \\ E_3 A_3^1 A_4^1 \widehat{A}_4^0 \widehat{A}_3^1 \widehat{E}_3 & (16) \end{cases}$$

We note that $(15) = \widehat{(16)}$:

$$\begin{cases} E_3 A_3^1 \widehat{A}_3^0 \widehat{E}_3 & (17) \\ E_3 A_3^0 A_4^1 \widehat{A}_4^0 \widehat{A}_3^1 \widehat{E}_3 & (18) \\ E_3 A_3^1 A_4^0 \widehat{A}_4^1 \widehat{A}_3^1 \widehat{E}_3 & (19) \end{cases}$$

(17)(18) = $\widehat{(19)}$:

$$\begin{cases} E_3 A_3^1 \widehat{A}_3^0 \widehat{E}_3 & (20) \\ E_3 A_3^1 A_4^1 \widehat{A}_4^0 \widehat{A}_3^1 \widehat{E}_3 & (21) \end{cases}$$

We are hence left with $2n^2$ equations in $5n^2$ unknowns (the elements of E_3, A_3^0 A_3^1, A_4^0 and A_4^1) and the resulting system is of the prescribed form.