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

Remi Geraud-Stewart$^{2,\ast}$,[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 $A := \{A_0, \ldots, A_{k-1}\}$, and a matrix $M$ guaranteed to be the product of some ordered subset of $L \subset A$, can $L$ be efficiently recovered? We begin by observing that the answer is positive under some assumptions on $A$.

Noting that appropriate transformations seem to make $L$’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 $A := \{A_0, \ldots, A_{k-1}\}$. For every permutation $\sigma$ of the set $\{0, \ldots, k-1\}$ let:

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

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

How hard the decomposition problem is depends on the choice of $p, n, k$, and $A$. 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 $A$ 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 $A$ (all other parameters being fixed) into a new set $\overline{A}$ makes that algorithm fail (see Section 3 for details). This yields the following two cryptosystem blueprints:

1. **Direct decomposition.** Define a set of matrices $A$ upon which decomposition is easy, and transform $A$ into $\overline{A}$ over which decomposition is hard. The public key is $\overline{A}$. Map the plaintext to a permutation $\sigma$ and define $\sigma \overline{A}$ as the ciphertext. To decrypt, transform $\sigma \overline{A}$ back into a $\sigma A$ which is easy to decompose over $A$ thereby yielding the message $\sigma$.

2. **Alternating decomposition.** Define two sets of matrices $A^b = \{A^b_0, \ldots, A^b_{k-1}\}$ with $b \in \{0, 1\}$ on which decomposition is easy\(^4\). Transform $A^b$ into public-key

\(^3\) $O(k^2 n^3 \log^* p)$ 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 \( \overline{A}^b \) over which decomposition is hard. Given a plaintext \( m \in \{0,1\}^k \), form the ciphertext:

\[
m\overline{A} := \prod_{i=0}^{k-1} A_i^m,
\]

where \( A := (A^0, A^1) \) and \( \overline{A} := (\overline{A}^0, \overline{A}^1) \). To decrypt, use the reverse transformation to obtain \( mA \), 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 \( A \) 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 \( A_0 \) or \( A_1 \). 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:\footnote{This conjecture is supported both by theoretical results [BG08, BS92, BKT04] and empirical evidence.}

\[5\]
**Conjecture 1.** Fix a security parameter $\lambda$. There exist values $p, k, n$ polynomial in $\lambda$ such that, if $A \subseteq \text{GL}_n(F_p)$ of size $k$ is sampled uniformly without replacement and $\sigma$ is a uniformly sampled permutation, no probabilistic algorithm can solve the decomposition problem of $\sigma A$ over $A$ with non-negligible probability in time polynomial in $\lambda$.

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. The size of an element $x \in F_p$ 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 $F_p$ will be referred to as elves. Formally, let:

$$
\mathcal{E} := \text{GL}_n(F_p) - Z(\text{GL}_n(F_p)),
$$

$$
\mathcal{D} := \{ M \in \mathcal{E} \text{ such that } |M| \leq \alpha \},
$$

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 $A \subseteq \mathcal{D}$. Then the largest coefficient in $\sigma A$ does not exceed $nk^{-1}\alpha k$. If $nk^{-1}\alpha k$ 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 $\sigma$ (see Remarks 1 and 6 for some precision about this).

---

**Input:** $p$ prime, $n > 1$, $M$ an $n \times n$ matrix over $F_p$, $A = \{A_0, \ldots, A_{k-1}\}$ invertible matrices over $F_p$

**Output:** $\{A_0, \ldots, A_{k-1}\}$ such that $M = \prod A_i$, or $\perp$

**Decompose**($A, M$):

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

---

**Fig. 1.** Polynomial-time decomposition algorithm over $A$ in $F_p$.

---

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 $A$ that doesn't belong to $D$, 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 $\sigma A$ over $\mathbb{F}_p$, for a uniformly sampled subset $A \subset \mathcal{C}$.

**Remark 1 (Incorrect guesses and backtracking).** Algorithm $\text{Decompose}$ assumes that only one matrix can satisfy the constraint; but because we work in $\mathbb{F}_p$, there is a non-zero probability $\eta$ 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 $\eta$ is small enough, as is the case for well-chosen parameters, the overall algorithm remains polynomial-time. Note that $\eta$ decreases rapidly at each algorithm step.

**Remark 2 (Complexity and memory).** If we ignore backtracking, assuming that elements in $A$ are provided together with their inverses, and that $\text{Decompose}$ completes without returning $\perp$, we can see that $\text{Decompose}$ claims $k(k - 1)/2$ matrix multiplications, and $n^2$ field element comparisons. The latter is negligible when compared to the former so $\text{Decompose}$’s overall complexity is dominated by $O(k^2)$ matrix products over $\mathbb{F}_p$: with textbook algorithms this is a total of $O(k^2n^3\log^2 p)$ time. Using efficient data structures, $\text{Decompose}$ 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 $\eta$. 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 $A = \{A_1, A_2, A_3\}$ with:

$$A_1 = \begin{pmatrix} 0 & 1 \\ 2 & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix}, \quad \text{and } \sigma A = \begin{pmatrix} 11 & 6 \\ 13 & 8 \end{pmatrix}.$$
We can now compute $A_i^{-1} \times \sigma A$ for $i = 1, 2, 3$ (decreasing entries are in blue, whereas increasing one are in red):

\[
A_1^{-1} \sigma A = \begin{pmatrix} 617283952 & 4 \\ 11 & 6 \end{pmatrix},
\]

\[
A_2^{-1} \sigma A = \begin{pmatrix} 3 & 2 \\ 4 & 2 \end{pmatrix},
\]

\[
A_3^{-1} \sigma A = \begin{pmatrix} 9 & 4 \\ 1234567884 & 1234567889 \end{pmatrix},
\]

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 $A_3$:

\[
A_1^{-1} (A_2^{-1} \sigma A) = \begin{pmatrix} 2 & 4 \\ 3 & 2 \end{pmatrix}, \quad A_3^{-1} (A_2^{-1} \sigma A) = \begin{pmatrix} 2 \\ 1234567890 & 1234567889 \end{pmatrix}.
\]

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

### 3 Trapdoor construction

Since Decompose works for $A \subset \mathcal{D}$ 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 $A \in \mathcal{A}$ 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 $D \in \mathcal{D}$ and $E \in \mathcal{E}$ be two secrets. For each $A_i \in \mathcal{A}$, let $\overline{A}_i := E A_i Y$, where $Y := DE^{-1}$.

   Indeed, with this choice, we have:

   \[
   \sigma \overline{A} = \prod_{i=0}^{k-1} \sigma A_i = \prod_{i=0}^{k-1} E A_{\sigma(i)} Y = E \left( \prod_{i=0}^{k-1} A_{\sigma(i)} D \right) E^{-1}.
   \]

   The trapdoor consists in computing $E^{-1}(\sigma \overline{A})ED^{-1}$, 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.

---

\* We do not recover exactly $\sigma A$ 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 $A_b^i \in A^b$ let $\overline{A}_i^b := E_i A_b^i E_{i+1}^{-1}$. With this choice, we have:

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

The trapdoor consists in computing $E_0^{-1}(m\overline{A})E_k$ which yields $mA$. 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 $A_i$. 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 $\overset{\$}{\leftarrow}$. We extend the latter notation to tuples, to mean sampling without replacement: for instance, $(x, y) \overset{\$}{\leftarrow} X$ does not have the same effect as $x \overset{\$}{\leftarrow} X$ followed by $y \overset{\$}{\leftarrow} X$. The symbol $\perp$ means “failure” and should be treated as an error condition.

Our algorithms are provided public parameters $pp = (n, p, k, \alpha)$, that have been chosen to meet a certain security level $\lambda$ (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 $A$. 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: $A_0^i$ versus $A_1^i$. Using these optimized versions of Decompose not only makes it faster, it also substantially reduces $\eta$ (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 $A$, and returning a subset of $A$ that can be found at this position:

- For the direct cryptosystem, $Sel(i, A)$ returns $A$ if $i$ is even, otherwise $\{D\}$;
- For the alternating cryptosystem, $Sel(i, A)$ returns $\{A_0^i, A_1^i\}$. 

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 $\text{Sel}(i, A)$: should the first one fail, we may assume that the other one passes (we don’t check it directly). For instance, given $M = A_1^1 A_0^1 A_2^0$, the first iteration computes $(A_1^0)^{-1} M$ and compares it to $M$ — the comparison fails and therefore $m_0 \neq 0$. Rather than confirming that $(A_1^0)^{-1} M \leq M$, we assume that it does, and directly pass to test:

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

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

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

and so forth. The matrices $(A_1^1 \ldots A_j^0 A_{j+1}^0)^{-1}$ can be precomputed. Note however that this version of $\text{Decompose}$ may never return $\perp$, even when given a product of matrices not in $A$, and therefore does not return a valid decomposition in all cases.

<table>
<thead>
<tr>
<th>D.KeyGen(pp):</th>
<th>D.Encrypt(pp, pk, $\sigma$):</th>
<th>D.Decrypt(pp, sk, C):</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $(A_0, \ldots, A_{k-1}, D) \leftarrow D$</td>
<td>1. $\overline{A} \leftarrow \text{pk}$</td>
<td>1. $(E, D, A) \leftarrow \text{sk}$</td>
</tr>
<tr>
<td>2. $E \leftarrow \mathcal{C}$</td>
<td>2. $C \leftarrow \sigma \overline{A}$</td>
<td>2. $T \leftarrow E^{-1} C E D^{-1}$</td>
</tr>
<tr>
<td>3. for $i = 0$ to $k - 1$:</td>
<td>3. return $C$.</td>
<td>3. $A' \leftarrow A \cup {D}^{k-1}$</td>
</tr>
<tr>
<td>$\overline{A}_i \leftarrow E A_i D E^{-1}$</td>
<td></td>
<td>4. $S \leftarrow \text{Decompose}(A', T)$</td>
</tr>
<tr>
<td>4. sk $\leftarrow (E, D, {A_i})$</td>
<td></td>
<td>5. if $S = \perp$ return $\perp$</td>
</tr>
<tr>
<td>5. pk $\leftarrow {\overline{A}_i}$</td>
<td></td>
<td>6. $(A_i, \ldots, A_{k-1}) \leftarrow S$</td>
</tr>
<tr>
<td>6. return sk, pk.</td>
<td></td>
<td>7. $\sigma \leftarrow (i_0, \ldots, i_{k-1})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8. return $\sigma$</td>
</tr>
</tbody>
</table>

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 $(p, n, k, \alpha)$ for each cryptosystem, as well as to the resulting key and ciphertext sizes as a function of the security level $\lambda$. 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_0^i, A_1^i) \leftarrow \mathcal{D})
   \text{with det } A_0^i = \text{det } A_1^i.
2. \((E_0, \ldots, E_k) \leftarrow \mathcal{E})
3. for i = 0 to k − 1:
   for b = 0 to 1:
   \quad A_i^b \leftarrow E_i A_i^b E_{i+1}^{-1}
4. sk \leftarrow (E_0, E_k, \{A_i^b\})
5. pk \leftarrow \{A_i\}
6. return sk, pk.

A.Encrypt(pp, pk, m):
1. $\overline{A} \leftarrow pk$
2. $C \leftarrow m \overline{A}$
3. return C.

A.Decrypt(pp, sk, C):
1. \((E_0, E_k, A) \leftarrow sk$
2. $T \leftarrow E_0^{-1} C E_k$
3. $T \leftarrow \text{Decompose}(A, T)$
4. if $T = \perp$ return \perp
5. \((A_0^{m_0}, \ldots, A_{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.

\begin{align*}
\text{DecomposeS}(A, M, i, \text{Sel}): & \\
1. \text{if } |A| = 0 \text{ return } \perp. \\
2. \text{if } |A| = 1 \text{ and } M \notin A \text{ return } \perp \text{ else return } A. \\
3. \text{for } A \in \text{Sel}(i, A), \\
\quad 3.1 M' \leftarrow A M.
\quad 3.2 \text{if } M' \leq M, \\
\quad \quad 3.2.1 L \leftarrow \text{DecomposeS}(A - \{A\}, M', i + 1, \text{Sel}). \\
\quad \quad 3.2.2 \text{if } L = \perp \text{ return } \perp \text{ else return } \{A\} \cup L. \\
4. \text{return } \perp
\end{align*}

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

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

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

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

Proof. The proof relies on the classical result that $Z(GL_n(\mathbb{F}_p)) = \mathbb{F}_p^* \mathbf{1}$ together with an adaptation of the fact that $GL_n(\mathbb{F}_p) = (p^n - 1)(p^n - p) \cdots (p^n - p^{n-1})$ 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 \(\alpha\) — 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:

Generic time-memory trade-offs. Being multiplicative, the proposed schemes lend themselves to generic time-memory trade-off attacks. We do not regard those as a significant threat given that increasing \(\lambda\) solves the problem. In such an attack the cryptanalyst partitions \(\text{pk}\) into two subsets \(\text{pk}_1\) and \(\text{pk}_2\) and forms two databases \(D_1, D_2\) each containing all possible products over the \(\text{pk}_i.\)

Because:

\[ C = D_1 D_2 \text{ for some } D_i = \text{product of elements } \in D_i \]

\[ D_1^{-1} C = D_2 \]

We can hence form the database \(E = D_1^{-1} C\) and look for collisions between \(E\) and \(D_2.\) This means in particular that if variants where the Hamming weight of the ciphertext is known to be small are used attacks exist. This is why we abstain from proposing such variants and insist on either a permutation or the alternating construction.

Algebraic relations. We adopt here an algebraic point of view to identify what information is available to an attacker through multiplicative manipulations of

---

\(^9\) This is not the norm induced on matrices by the infinity norm on vectors, and in particular it is not submultiplicative.
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 $A_i$, 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 $A_i^s$, if there aren’t too many variables. Independently, an attacker successfully recovering a product of the form $A_iA_j^{-1}$ for any two indices $i, j$ can then 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:

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

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 the 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 $\alpha$ values. 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.

\footnote{Or equivalently $A_iD^{-1}$ for any index $i$.}
\footnote{For $\alpha = 1$ and $n = 9$, the distribution of dwarf determinants has a Shannon entropy of 4 bits, regardless of $p$.}
**Determinant ratio attacks.** Consider two public-key elements in the alternating cryptosystem $A_0^i, A_1^i$ at some rank $i$ and let:

$$\Delta = \frac{\det A_1^i}{\det A_0^i} \mod p = \frac{\det A_1^i}{\det A_0^i} \mod p$$

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

$$\Delta \cdot \det A_0^i = \det A_1^i$$

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 $\Xi$ be the ratio between the information consumed to produce that key and the size of the public key, i.e.:

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

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

The new cryptosystems have respectively:

$$\Xi_D = \frac{(k + 1) \log_2 \beta(n, 1 + \alpha) + \log_2 \beta(n, p)}{k \log_2 \beta(n, p)}$$

$$\Xi_A = \frac{2k \log_2 \beta(n, 1 + \alpha) + (k + 1) \log_2 \beta(n, p)}{2k \log_2 \beta(n, p)}$$

Note that $\Xi_A > \Xi_D$ which seems to indicate that $A$ is a seemingly “stronger” variant as per this criterion. The effect of $\alpha$ is negligible at practical security levels $\lambda$.

**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

$$S = BMT = \text{diag}(\gamma_1, \gamma_2, \ldots, \gamma_r, 0, \ldots, 0)$$
with $r$ is the matrix rank\textsuperscript{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 $\gamma_i$ 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.

### 5.3 Parameters for D

The parameters $(p, n, k, \alpha)$ must satisfy several constraints:

- Brute-force exhaustion of the ciphertext should be intractable: $k! \geq 2^\lambda$.
- Encryption must be injective, which implies that $|\mathcal{E}| > k!$.
- 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: $|\mathcal{D}| > 2^\lambda$.
- Decomposition works under the assumption that $\|E^{-1}(\sigma A)DE^{-1}\|_\infty < p$, which is guaranteed by Corollary 1 if $\alpha 2^k n^{2k-1} < p^{13,14}$.

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 $\alpha$, then select the values such that $n\alpha$ is smallest; then fix $k$ and select the smallest prime $p$ bigger than $\alpha 2^k n^{2k-1}$. Then, by design, all the above constraints are satisfied\textsuperscript{15}.

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

\textsuperscript{13} Note that neither of these conditions is necessary, in light of Remark 6, although we treat them as such.

\textsuperscript{14} This condition must be adapted in case we select the $D = 1$ variant discussed in Remark 4.

\textsuperscript{15} Additional parameter sets, complete with worked out examples, can be found in Appendix D.
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 $|\mathcal{E}| > 2^k$.
- 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: $|\mathcal{D}| > 2^\lambda$.
- Decompose works under the assumption that $\|E_{\mathbf{0}}^{-1}(m\mathbf{X})E_k\|_\infty < p$, which is guaranteed by Corollary 1 if $\alpha^k n^{k-1} < p^{16}$.

Finding $n$ and $\alpha$ 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 $2^k$ dwarves and two elves, the public key consists in $2^k$ elves, the ciphertext is an elf.

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

Here again, Remark 6 indicates that this choice is overly conservative.
Table 2. Suggested parameter sets for $A$.

<table>
<thead>
<tr>
<th>Version</th>
<th>Toy</th>
<th>Challenge</th>
<th>Recommended</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>16</td>
<td>64</td>
<td>128</td>
<td>512</td>
</tr>
<tr>
<td>$k$</td>
<td>16</td>
<td>64</td>
<td>128</td>
<td>512</td>
</tr>
<tr>
<td>$n$</td>
<td>4</td>
<td>8</td>
<td>10</td>
<td>24</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$p$</td>
<td>$2^{17} + 5$</td>
<td>$2^{115} - 19$</td>
<td>$2^{573} + 549$</td>
<td>$2^{2859} + 641$</td>
</tr>
<tr>
<td>$pk$ (B)</td>
<td>3072</td>
<td>261 120</td>
<td>1 772 800</td>
<td>210 862 080</td>
</tr>
<tr>
<td>$sk$ (B)</td>
<td>320</td>
<td>6128</td>
<td>20 250</td>
<td>559 296</td>
</tr>
<tr>
<td>$C$ (B)</td>
<td>96</td>
<td>2040</td>
<td>6925</td>
<td>205 920</td>
</tr>
</tbody>
</table>

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. In particular, it is possible that the new schemes lend themselves to some natural variant of [BCJ11] but even if things are so, this would not be a problem as the problem would remain exponential in nature and hence fixable by increasing parameters.

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\textsuperscript{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 $A_i$s 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!).

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

\textsuperscript{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].

\textsuperscript{19} Neither OAEP nor its variants are quantum secure, unless appropriate modifications are applied, cf. [Ebr22].
We suggest the following idea applicable to the alternating version. Instead of encoding only one bit using either $A_0^i$ or $A_1^i$, pick for each position $i$ a family of $u$ pairs of scalars:

$$(\tau_{v_i,a}, \rho_{v_i,a}), (\tau_{v_i,1}, \rho_{v_i,1}), \ldots, (\tau_{v_{i,u-1}}, \rho_{v_{i,u-1}})$$

such that $\forall j$, $\gcd(\tau_{v_{i,j}}, \rho_{v_{i,j}}) = 1$.

Use a different linear combination $\tau_{v_{i,u-1}} A_0^i + \rho_{v_{i,u-1}} A_0^i$ 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 $1061 \approx 2^{10}$ 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 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 $p$.

<table>
<thead>
<tr>
<th>$\phi$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>10</th>
<th>14</th>
<th>20</th>
<th>29</th>
<th>41</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\eta$</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>32</td>
<td>64</td>
<td>128</td>
<td>256</td>
<td>512</td>
<td>1024</td>
</tr>
</tbody>
</table>

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

**Key generation.** A central assumption underlying our schemes is that private keys allowing decryption can be efficiently generated. The exact requirements are 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 $\alpha$ small enough to make finding a valid key impossible (see Appendix C). Therefore the following questions are open: under what exact conditions on $A$ is $\sigma \mapsto \sigma A$ (resp. $m \mapsto m A$) 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 $A_0^4, A_1^4, A_0^5, A_1^5 \in \mathcal{D}$, used to encode two successive plaintext bits (say bits 4 and 5) as in the alternating construction. We wish all four products $A_i^4 A_j^5$ for $(i, j) \in (0,0), (0,1), (1,0), (1,1)$ to be simultaneously dwarves.

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

$$A_i^4 A_j^5 = M_{4,5}^{i,j} \mod p.$$
Here, there are $4n^2$ unknowns in $4n^2$ equations; these unknowns are the coefficients of $A^0_1, A^1_1, A^0_5$ and $A^1_5$. We can solve it exactly, and having solved those equations we repeat the process for matrices at other ranks. Upon encryption we now have:

$$m\overline{X} = E_0(A^0_0 A^1_1) \cdots (A^0_{k-2} A^1_{k-1}) E^{-1}_k$$

$$= E_0 M^0_{0,1} \cdots M^0_{k-2,k-1} E^{-1}_k$$

$$= E_0 (m\overline{X}) E^{-1}_k$$

where each matrix $M^{a,b}_{i,j} \in \mathcal{D}$ 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 $\sigma_1 A \sigma_2 A \neq (\sigma_1 \sigma_2) A$ in general. Yet there is a natural map sending $(\sigma_1, \sigma_2)$ to a permutation $\tau$ over $\mathbb{A}^2 := \mathbb{A} \cup \mathbb{A}$ such that $\sigma_1 A \sigma_2 A = \tau \mathbb{A}^2$. 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 $V^b_i := S_i A^b_i S_{i+1} \mod t$ where the $S_i$ are new random elves. Let $V$ denote the set of the $V^b_i$. Encapsulation is performed by computing $C = m\overline{X}$ for a randomly chosen $m$ to get the ciphertext, and $K = H(mV \mod t)$ to get the shared key, where $H$ is a hash function. Decapsulation\(^{21}\) computes:

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

**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.\(^{20}\) The same idea can be adapted to the direct cryptosystem.\(^{21}\) Note that this variant does not use Decompose and that decapsulation is very fast and that the decapsulation key footprint is very low.
However our attempts to do so have all resulted in reduced $\eta$ 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 $\text{pk}$ 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 reduce keysize. The entropy loss caused by such variants might affect both security and $\eta$ 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.

To track precisely the evolution of attacks to come, it would be great if future cryptanalysts could continue adopting the notations $n, k, p, \alpha, d$ and precise in their papers to which parameter combinations their attacks apply as well as the concerned variant (direct, alternating, KEM) and its mode of operation (bare encryption which is not recommended or the semantic transform).
References


HS03. Dennis Hofheinz and Rainer Steinwandt. A practical attack on some braid group based cryptographic primitives. In Yvo Desmedt, editor, Public Key
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 $F$, 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 $H, H'$, 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.

![Diagram](image)

**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:

$$[a_0, a_1, \ldots, a_r] := A_0^{a_0} A_1^{a_1} \cdots A_r^{a_r}.$$  

Let’s first establish the base case: the value $E_0 [1] (E_0 [0])^{-1}$ is public. Indeed, this is exactly $A_1^0 (A_0^1)^{-1}$. 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:

\[
I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad M_1 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad M_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \\
J = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad M_3 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \quad M_3^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}
\]

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

\[
(M_1 M_2)^{-1} = (J M_1)^{-1} = M_2 M_1 = J M_2 = M_3 \\
M_3 M_3^{-1} = M_1^2 = M_2^2 = J^2 = M_3^3 = I
\]

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.

22
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:

\[
\mathcal{A}_0 = \begin{pmatrix} 235 & 402 \\ 479 & 659 \end{pmatrix}, \quad \mathcal{A}_1 = \begin{pmatrix} 416 & 510 \\ 608 & 484 \end{pmatrix}, \quad \mathcal{A}_2 = \begin{pmatrix} 36 & 846 \\ 845 & 856 \end{pmatrix}.
\]

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

\[
C = \sigma \mathcal{A} = \mathcal{A}_1 \mathcal{A}_0 \mathcal{A}_2 = \begin{pmatrix} 521 & 99 \\ 347 & 464 \end{pmatrix}.
\]

**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 \( A_0, A_1, A_2 \):

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

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

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. \( (i = 1, \text{branch 0}) \) 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. \((i = 1, \text{branch 1})\) 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 \(A_0, A_2\):

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

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:

\[
A_0^0 = \begin{pmatrix} 24 & 94 \\ 91 & 45 \end{pmatrix}, \quad A_0^1 = \begin{pmatrix} 63 & 51 \\ 26 & 57 \end{pmatrix}, \quad A_0^2 = \begin{pmatrix} 85 & 42 \\ 16 & 15 \end{pmatrix}, \quad A_0^3 = \begin{pmatrix} 44 & 29 \\ 49 & 12 \end{pmatrix}
\]

\[
A_1^0 = \begin{pmatrix} 27 & 23 \\ 41 & 22 \end{pmatrix}, \quad A_1^1 = \begin{pmatrix} 41 & 36 \\ 70 & 38 \end{pmatrix}, \quad A_1^2 = \begin{pmatrix} 73 & 69 \\ 87 & 47 \end{pmatrix}, \quad A_1^3 = \begin{pmatrix} 52 & 33 \\ 80 & 67 \end{pmatrix}
\]

\[
E_0 = \begin{pmatrix} 181465853 & 489998889 \\ 586318319 & 354105151 \end{pmatrix}, \quad E_1 = \begin{pmatrix} 737027227 & 204799375 \\ 240734459 & 201952588 \end{pmatrix}
\]

\[
E_2 = \begin{pmatrix} 462605764 & 533199356 \\ 028731110 & 540728462 \end{pmatrix}, \quad E_3 = \begin{pmatrix} 153582549 & 270321504 \\ 280451465 & 475609880 \end{pmatrix}
\]

\[
E_4 = \begin{pmatrix} 493998496 & 252943122 \\ 113547886 & 273933344 \end{pmatrix}
\]

The public key is therefore:
Consider the plaintext \( m = (1, 0, 1, 0) \). The corresponding ciphertext is:

\[
C = X_0^1 X_1^0 X_2^1 X_3^0 = \begin{pmatrix}
736131904 & 41083532 \\
281879757 & 104491974
\end{pmatrix}
\]

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 \( A_1^3 \):

\[
T_3(A_3^0)^{-1} = \begin{pmatrix}
401683 & 284967 \\
521330 & 374910
\end{pmatrix} \leq T_3(A_1^3)^{-1} = \begin{pmatrix}
748950526 & 383577638 \\
363305113 & 24238044
\end{pmatrix}
\]

\[
\downarrow
\]

\[
T_2 = T_3(A_3^0)^{-1}
\]

\[
T_2(A_2^0)^{-1} = \begin{pmatrix}
744281048 & 796032103 \\
553236859 & 210961813
\end{pmatrix} \geq T_2(A_2^1)^{-1} = \begin{pmatrix}
2299 & 2688 \\
3155 & 3345
\end{pmatrix}
\]

\[
\downarrow
\]

\[
T_1 = T_2(A_2^1)^{-1}
\]

\[
T_1(A_0^0)^{-1} = \begin{pmatrix}
27 & 23 \\
41 & 22
\end{pmatrix} \geq T_1(A_1^1)^{-1} = \begin{pmatrix}
761746477 & 330977107 \\
390852515 & 113929291
\end{pmatrix}
\]

\[
\downarrow
\]

\[
T_0 = T_1(A_0^0)^{-1} = A_0^1 \Rightarrow m = (1, 0, 1, 0)
\]

We instrument this example to also illustrate the leakage through the determinant addressed in the paper. We purposely took in the example dwarves \( A_0^0, A_1^1 \) having nonidentical determinants. Compute for instance:
\[
\Delta = \frac{\det \begin{bmatrix} 1 \\ 0 \end{bmatrix}}{\det \begin{bmatrix} 0 \\ 0 \end{bmatrix}} \mod p = \frac{\det A_0^1}{\det A_0^0} \mod p = 213112125
\]

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 \(\rho\) as a modular ratio and reveal the determinants of the \(A_0^b\). There are \(103020000 \approx 2^{26.61}\) invertible matrices with coefficients bounded by 100. All matrix pairs whose \(\Delta = 213112125\) have determinants equal to \(\pm \{349 \Delta, 7474\} \mod p\). Each of the two pairs appears \(9702 \approx 2^{13.24}\) 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.:

\[
\{\begin{bmatrix} A_0^0 \\ A_1^0 \end{bmatrix}, \begin{bmatrix} A_0^1 \\ A_1^1 \end{bmatrix}\} = \{E_3 A_0^0 \widehat{E}_4, E_3 A_1^0 \widehat{E}_4\}
\]

Form \(\begin{bmatrix} \alpha_3 \\ \alpha_4 \end{bmatrix} = E_3 A_0^0 \widehat{E}_4 E_4 A_1^0 \widehat{E}_3 = E_3 A_0^0 A_1^0 \widehat{E}_3 \widehat{E}_4\); 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 \(A_0^0\) and \(A_1^0\)).

Take two successive public-key elements for instance at rank 3 and 4 and form the 4 quantities \(S_{b_3, b_4} = \begin{bmatrix} \alpha_3 \\ \alpha_4 \end{bmatrix} = E_3 A_0^0 A_1^0 \widehat{E}_4 E_4 A_0^b \widehat{E}_4 = E_3 A_1^0 A_0^b \widehat{E}_3 \widehat{E}_4\).

At a first resolution step we eliminate \(E_3\). There are 16 ways to do so (of which 4 are trivial that we remove):

\[
\begin{align*}
S_{0,0} \widehat{S}_{0,1} & \quad S_{0,0} \widehat{S}_{1,0} & \quad S_{0,0} \widehat{S}_{1,1} \\
S_{0,1} \widehat{S}_{0,0} & \quad S_{0,1} \widehat{S}_{1,0} & \quad S_{0,1} \widehat{S}_{1,1} \\
S_{1,0} \widehat{S}_{0,0} & \quad S_{1,0} \widehat{S}_{0,1} & \quad S_{1,0} \widehat{S}_{1,1} \\
S_{1,1} \widehat{S}_{0,0} & \quad S_{1,1} \widehat{S}_{0,1} & \quad S_{1,1} \widehat{S}_{1,0}
\end{align*}
\]

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

\[
\left\{ \begin{array}{c}
E_3 A_0^0 \widehat{E}_3 \\
E_3 A_0^0 A_1^0 \widehat{E}_3 \\
E_3 A_1^0 \widehat{E}_3 \\
E_3 A_0^0 A_1^0 A_0^0 \widehat{E}_3 \\
E_3 A_1^0 A_1^0 A_0^0 \widehat{E}_3 \\
E_3 A_1^0 A_1^0 A_1^0 \widehat{E}_3 
\end{array} \right\}
\]
Multiplying (2),(3) and (5) by (1) we get:

\[
\begin{align*}
E_3 A_3^0 A_3^0 & \cdot E_3 \\
E_3 A_3^0 A_4^0 A_3^0 & \cdot E_3 \\
E_3 A_3^0 A_4^0 A_4^0 & \cdot E_3 \\
E_3 A_3^1 A_3^0 A_3^0 & \cdot E_3 \\
E_3 A_3^1 A_3^0 A_4^0 & \cdot E_3 \\
E_3 A_3^1 A_4^0 A_3^0 & \cdot E_3 \\
E_3 A_3^1 A_4^0 A_4^0 & \cdot E_3 \\
E_3 A_3^1 A_4^1 A_3^0 & \cdot E_3 \\
E_3 A_3^1 A_4^1 A_4^0 & \cdot E_3 \\
E_3 A_3^1 A_4^1 A_4^1 & \cdot E_3 \\
\end{align*}
\]

Eliminating duplicates we get:

\[
\begin{align*}
E_3 A_3^1 A_3^0 & \cdot E_3 \\
E_3 A_3^0 A_3^1 A_3^0 & \cdot E_3 \\
E_3 A_3^0 A_3^0 A_4^0 & \cdot E_3 \\
E_3 A_3^1 A_3^0 A_4^0 & \cdot E_3 \\
E_3 A_3^1 A_4^0 A_3^1 & \cdot E_3 \\
\end{align*}
\]

We note that (15) = (16):

\[
\begin{align*}
E_3 A_3^1 A_3^0 & \cdot E_3 \\
E_3 A_3^0 A_3^1 A_3^0 & \cdot E_3 \\
E_3 A_3^0 A_3^0 A_4^0 & \cdot E_3 \\
\end{align*}
\]

(17)(18) = (19):

\[
\begin{align*}
E_3 A_3^1 A_3^0 & \cdot E_3 \\
E_3 A_3^1 A_4^0 A_3^1 & \cdot E_3 \\
\end{align*}
\]

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.