The **ALTEQ** Signature Scheme: Algorithm Specifications and Supporting Documentation

Markus Bläser[1], Dung Hoang Duong[3], Anand Kumar Narayanan[4], Thomas Plantard[5], Youming Qiao[2], Arnaud Sipasseuth[6], and Gang Tang[2]

[1] Department of Computer Science, Saarland University, Saarland Informatics Campus, Saarbrücken, Germany.
[2] Centre for Quantum Software and Information, School of Computer Science, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW, Australia.
[3] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia.
[4] SandboxAQ, Palo Alto, United States.
[5] Nokia Bell Labs, Murray Hill, New Jersey, United States.
[6] Information Security Laboratory of KDDI Research, Inc, Fujimino, Saitama, Japan

Document Version 2023-09-15.

Version History

Version 1.0.0 2023-06-01

- Initial version.

Version 1.0.1 2023-09-14

1. Adding a machine to test performance; see Section 5.2.
2. Fixing typos.

Version 1.1 2023-09-15

1. Fig 2: In Vf(), we add a function checkInvertibility to verify that the diagonal elements in column matrices are non-zero to guarantee invertibility so as to prevent the attack proposed by Markku-Juhani O. Saarinen.
2. Section 2.4: delete the function actingOnATFSwTensor, add two functions columnsMatrix and columnsDecomposition.
3. Fig 2: In Sign(), we submit column matrices $\{D_i^{col}\}_{i \text{ s.t. } c_i \neq C}$ instead of normal invertible matrices $\{D_i\}_{i \text{ s.t. } c_i \neq C}$.
4. Fig 2: in Vf(), we remove actingOnATFSwTensor and replace it with actingOnATFS when $c_i \neq C$.
5. Because of (2),(3) and (4), there is a performance improvement in the code speeds. Please see Section 1.3, Section 5.2 for the updated performance.
6. Fixing typos.

Table of Contents

1 Introduction

We present the digital signature scheme ALTEQ based on the hardness of the Alternating Trilinear Form Equivalence (ATFE) problem.

The construction has two steps. First, following the Goldreich–Micali–Wigderson (GMW) motif [GMW91], we devise a zero-knowledge protocol reliant on the hardness of ATFE. We then apply the Fiat–Shamir (FS) transformation [FS86] to remove the interaction from the zero-knowledge protocol and obtain a digital signature scheme. An overview of these steps will be presented in Section 1.1.

Some key considerations for ALTEQ are as follows.

Simple and standard protocol design. We use the standard Goldreich–Micali–Wigderson protocol [GMW91] and Fiat–Shamir transformation [FS86]. Both GMW and FS are classical and their securities are well-understood in classical and quantum random access models [KLS18,LZ19,DFMS19]. For example, the QROM security of ATFE-GMW-FS can be proved via two approaches [BCD+22] (based on certain assumptions on the ATFE problem), one of which gives a tight security reduction.

Support for ATFE: post-quantum, complexity-theoretic, and practical. Our choice of ATFE is based on several theoretical and practical considerations.

1. From the quantum algorithm viewpoint, there is a strong negative evidence for the standard approach of the hidden subgroup problems to work for general linear groups over finite fields [HMR+10]. This could be considered as an inherent bottleneck for current quantum algorithms to work against the general linear hidden subgroup problem. Such an argument is applicable to ATFE as it relies on a group action by general linear groups over finite fields.
2. From the computational complexity viewpoint, the recent complexity theory of tensor isomorphism [GQ21b,GQT21] reveals that ATFE is polynomial-time equivalent to many algebraic isomorphism problems, including tensor isomorphism, matrix code equivalence, p-group isomorphism, and polynomial isomorphism. The monomial code equivalence reduces to ATFE in polynomial time [GQ21a].
   These problems have been studied in several research communities, such as coding theory, cryptography, theoretical computer science, and computational group theory. Despite research efforts from these communities, these problems are regarded as difficult to solve in practice. See Appendix A.1 for more details.
3. From the practical algorithm viewpoint, the connections with other problems allows us to tap into the pool of practical algorithms for polynomial isomorphism [BFV13], matrix code equivalence [CNP+22], group isomorphism [LQ17], and min rank [BBC+20]. Recent new algorithms for ATFE [Qia23,Beu22] are built on works along these lines. From this perspective, we could say that the practical hardness of ATFE has been studied in several communities for quite some time.

1.1 Overview of the basic approach

This protocol consists of two steps. First, apply the Goldreich–Micali–Wigderson (GMW) protocol to ATFE to obtain an identification (or Sigma) protocol. Second, apply the Fiat–Shamir (FS) transformation to the identification protocol.

For a clear overview, we introduce the GMW-FS design, and the ATFE problem, separately. In particular, for the GMW-FS design, we shall introduce it first using an abstract group action.

The basic GMW-FS design. The GMW-FS design takes a group action and gives a digital signature scheme.

Group action. Let G be a finite group, S a finite set, and $\alpha : G \times S \to S$ be a group action. We assume that group and set elements have efficient representations in algorithms, α can be computed efficiently, and elements of G and S can be sampled in uniform random efficiently.

Notation. For $n \in \mathbb{N}$, $[n] := \{1, 2, \ldots, n\}$. The notation $\leftarrow_R$ denotes uniform random sampling; for example, $g \leftarrow_R G$ denotes that g is sampled in uniform random from G.

Parameters for the basic GMW-FS design. The following system parameters are used.

1. $C = 2^c$: The number of set elements as the public key, and the number of group elements as the private key.
2. r: The number of rounds in the scheme.

Key generation.

1. $s_1 \leftarrow_R S$.
2. $g_1 :=$ Id, the identity element in the group G.
3. $g_2, \ldots, g_C \leftarrow_R G$
4. For $i = 2, \ldots, C$, $s_i := \alpha(g_i, s_1)$.
5. Public key is $(s_1, \ldots, s_C) \in S^C$.
6. Private key is $(g_1, \ldots, g_C) \in G^C$.

Signing. Let M be the message to be signed. Let $H : \{0,1\}^* \to \{0,1\}^\ell$ be a hash function, where $\ell = r \cdot c$.

1. For $i \in [r]$, $h_i \leftarrow_R G$. Let $t_i := \alpha(h_i, s_1)$.
2. Let $L := H(M \mid t_1 \mid \cdots \mid t_r) \in \{0,1\}^\ell$.
   Slice L into r length-c bit strings, i.e. $L = b_1 \mid \cdots \mid b_r$, where $b_i \in \{0,1\}^c$.
3. For $i \in [r]$, let $f_i := h_i \cdot g_{b_i}^{-1}$.
4. The signature is $(b_1, \ldots, b_r, f_1, \ldots, f_r)$.

   Note that $\alpha(f_i, s_{b_i}) = \alpha(h_i \cdot g_{b_i}^{-1}, s_{b_i}) = \alpha(h_i, \alpha(g_{b_i}^{-1}, s_{b_i})) = \alpha(h_i, s_1) = t_i$.

Verification. The verifier receives the message M and the signature $(b_1, \ldots, b_r, f_1, \ldots, f_r)$.

1. For $i \in [r]$, let $t'_i := \alpha(f_i, s_{b_i})$.
2. Let $L' := H(M|t'_1|\ldots|t'_r)$.
3. Accept if L' is equal to $L = b_1 \mid \cdots \mid b_r$. Reject otherwise.

The group action underlying ATFE. The ALTEQ scheme is obtained by instantiating the group action $\alpha : G \times S \to S$ as follows. See also Section 2.3 for more details.

Parameters for the ATFE group action.

1. n: the vector space dimension.
2. q: the finite field order.

The ATFE group action definition.

1. The group G is $\mathrm{GL}(n, q)$, the general linear group over the finite field of order q.
2. The set S is the set of alternating trilinear forms $\mathrm{ATF}(n, q) := \{\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q\}$, where ϕ is trilinear (linear in each argument) and alternating (ϕ evaluates to 0 whenever two arguments are the same).
3. The action α is defined as follows. For $A \in \mathrm{GL}(n, q)$ and $\phi \in \mathrm{ATF}(n, q)$, $\phi \circ A$ is an alternating trilinear form defined by $(\phi \circ A)(u, v, w) = \phi(A^{\mathrm{t}}(u), A^{\mathrm{t}}(v), A^{\mathrm{t}}(w))$.

1.2 The ALTEQ scheme

The ALTEQ scheme implementation incorporates several measures to enhance the system performance. Some main points are as follows.

Unbalanced challenges. We incorporate the *unbalanced challenge* technique [FS86]. Briefly speaking, this means that in the GMW identification protocol, we set a fixed number of challenges to be some specific value. This is because when the challenge is of this value, the response is a random matrix expanded from a short seed, so sending this seed through reduces the communication (and thus the signature size). The cost is that more rounds are required, therefore increasing the sign and verification times.

Implementation considerations. The main algebraic operation is the group action computation, which relies on modular arithmetic. For modular arithmetic, we use a method for Pseudo-Mersenne numbers from [Cra92]. For group actions, we implement several optimisations, such as the tensorial viewpoint of alternating trilinear forms, and the use of decomposing an invertible matrix into a product of matrices in a special form.

The security of the GMW-FS design. The security of the GMW-FS design of digital signatures is well-understood in both the Random Oracle Model (ROM) and the Quantum Random Oracle Model (QROM) models. The existential unforgeability under chosen-message attack (EUF-CMA) security is well-known assuming some hardness notion of group actions; a concrete treatment for ATFE was presented in [TDJ+22]. The EUF-CMA security in QROM was shown in [BCD+22] based on [KLS18,LZ19,DFMS19] via the perfect unique response and lossy properties.

Parameter choices. Let λ be the bit security level. To determine the choices of n and q (the ATFE parameters), we rely on two main approaches for solving ATFE: the Gröbner basis approach and the approach based on low-rank points. The Gröbner basis approach determines the vector space dimension n, and then the low-rank based approach determines the field order q. The GMW-FS design parameters, namely the round number r and the form number C, and the unbalanced challenge parameter K, can be determined in a straightforward manner. There can be certain flexibility in getting some trade-offs between signature and public key sizes, as well as key generation, sign, and verify times.

1.3 An overview of parameters and performance of ALTEQ

We have parameter sets I and III of ALTEQ aimed for NIST security categories I/II and III/IV respectively. We also *suggest*[7] parameter set V aimed for NIST security category

[7] See Section 1.4 for a discussion on why this is just a suggestion.

V. We consider two groups of parameters, one for small public key+signature sizes (called Balanced), and the other one for short signatures (called ShortSig).

We tested the codes on a server with CPU as Intel Xeon E-2288G 3.7GHz 8 cores 16MB L3 Cache HT Enabled (Max Turbo Freq. 5.0GHz, Min 4.7GHz), 64GB memory, and on Red Hat Enterprise Linux 8.6. The codes are compiled by gcc version 8.5.0.

An overview of the results can be found in Table 1. The sizes in Table 1 are measured in KiloBytes (KB, 1 KB=1024 bytes), and the timings are measured in millisecond (ms, 1 ms=0.001 second). For more detailed information, please see Section 5.

parameter set	mode	public key size (KB)	signature size (KB)	key generation (ms)	signature generation (ms)	signature verification (ms)
I	Balanced	8	16	0.093	0.629	0.496
	ShortSig	512	10	1.902	0.194	0.092
III	Balanced	32	48	0.582	6.986	6.483
	ShortSig	1024	24	5.152	1.705	1.304
V	Balanced	72	128	1.531	11.707	10.351
	ShortSig	2048	64	12.567	6.894	5.969

Table 1: An overview of the parameters and performance of ALTEQ.

1.4 Remarks, advantages, and limitations

Variations of the parameters. The parameters in Table 1 can be tuned to allow for some trade-offs between the parameters. For example, for the balanced level I option, it is possible to gain approximately 25% speed-up at the cost of increasing the public key+signature sizes by 4KB.

It is also possible to further speed-up the codes, and optimize the memory costs *during* the executions of the sign and verify procedures.

Reducing the signature sizes. The signature sizes of ALTEQ can be reduced at the expenses of increasing the signing and verification times, if one of the following two techniques is used.

The first one is called the seed tree. The seed tree is used to generate the challenges. It starts with the root and use a pseudorandom generator to generate the challenges via a tree structure. A version of ALTEQ with seed trees has been implemented, but it is not included in this submission, as more experiments will be needed to examine the trade-offs it brings.

The second one is called the multiparty computation in the head (MPC-in-the-head) protocol. It is a classical technique and recently shown to work for the GMW-FS design in general [Jou23]. It remains to investigate into this optimization technique; a preliminary study can be found in [BCD$^+$22].

Ring signatures. The GMW-FS design supports the linkable ring signature functionality following [BKP20], and a preliminary implementation based on ALTEQ was reported in [BCD$^+$22]. This would allow a signer can sign on behalf of a group chosen by him-or-herself, while retaining anonymous with in the group without a complex setup procedure or the requirement for a group manager. The linkable property ensures that signatures produced by the same signer can be publicly linked.

Advantages. We've mentioned some advantages, such as simple and standard protocol design, supporting ring signature functionality, and some flexibility in variations of the parameters. The choice of ATFE in post-quantum cryptography is backed by a strong limitation on known quantum algorithm techniques [HMR+10]. The speeds of our implementation, though still slower than lattice-based schemes, are acceptable in general.

Limitations. Our public key and signature sizes are still relatively large, especially for levels III and V. There are also some research questions for better understanding the effects of the direct Gröbner basis attacks (see Appendix A.2 and also Footnote 15). This is the main reason why the parameters for level V are only a suggestion, as going to larger n may be needed.

2 Basic operations

2.1 Basic notations

We list some common notations.

1. $\mathbb{F}_q$, q a prime power: the finite field of order q.
2. $\mathbb{F}_q^n$: the vector space of length-n column vectors over $\mathbb{F}_q$.
3. $\mathrm{M}(n,q)$: the linear space of $n \times n$ matrices over $\mathbb{F}_q$.
4. $\mathrm{GL}(n,q)$: the general linear group of degree n over $\mathbb{F}_q$.
5. For $u \in \mathbb{F}_q^n$ and $A \in \mathrm{M}(n,q)$, u^{t} and A^{t} denote their transposes.
6. For a real number a, let $\lceil a \rceil$ (resp. $\lfloor a \rfloor$) denote the smallest (resp. largest) integer that is greater (resp. smaller) than a. Denote by $\lfloor a \rceil$ the integer closest to a.
7. For two non-negative integers a, b, let $\binom{a}{b}$ be the binomial coefficient.
8. For a positive integer m, let $[m] := \{1, 2, \ldots, m\}$.
9. For a finite set S, let $a \in_R S$ mean that a is a uniformly random sample from S.

2.2 Trilinear forms and a natural group action on them

A trilinear form on $\mathbb{F}_q^n$ is a map $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$ that is $\mathbb{F}_q$-linear in each argument. Such a trilinear form is called *alternating* if and only if ϕ evaluates to 0 whenever two arguments are the same, i.e., for $u, v \in \mathbb{F}_q^n$, we have $\phi(u,u,v) = \phi(u,v,u) = \phi(v,u,u) = 0$. Denote by $\mathrm{ATF}(n,q)$ the linear space of all alternating trilinear forms on $\mathbb{F}_q^n$.

The general linear group $\mathrm{GL}(n,q)$ naturally acts on $\mathrm{ATF}(n,q)$ as follows: $A \in \mathrm{GL}(n,q)$ sends ϕ to $\phi \circ A$, defined as $(\phi \circ A)(u,v,w) := \phi(A^{\mathrm{t}}(u), A^{\mathrm{t}}(v), A^{\mathrm{t}}(w))$ for all $u, v, w \in \mathbb{F}_q^n$. This action defines an equivalence relation $\cong$ on $\mathrm{ATF}(n,q)$, namely $\phi \cong \psi$ if and only if there exists $A \in \mathrm{GL}(n,q)$ such that $\phi = \psi \circ A$.

2.3 Alternating trilinear forms and group actions in algorithms

Representing an alternating trilinear form in algorithms. An alternating trilinear form $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$ can be uniquely represented as $\sum_{1 \le i < j < k \le n} c_{i,j,k} e_i^* \wedge e_j^* \wedge e_k^*$, where $c_{i,j,k} \in \mathbb{F}_q$, e_i^* is the linear form sending $u = (u_1, \ldots, u_n)^{\mathrm{t}} \in \mathbb{F}_q^n$ to u_i, and $\wedge$ denotes the wedge (or exterior) product. The $e_i^* \wedge e_j^* \wedge e_k^*$ as an alternating trilinear form sends

$$(u, v, w) \longmapsto \det \begin{bmatrix} u_i & v_i & w_i \\ u_j & v_j & w_j \\ u_k & v_k & w_k \end{bmatrix},$$

where $u = (u_1, \ldots, u_n)^{\mathrm{t}}$, $v = (v_1, \ldots, v_n)^{\mathrm{t}}$, $w = (w_1, \ldots, w_n)^{\mathrm{t}}$ are in $\mathbb{F}_q^n$. Therefore, in algorithms we can store the alternating trilinear form ϕ as

$$(c_{i,j,k} : 1 \le i < j < k \le n), c_{i,j,k} \in \mathbb{F}_q,$$

which requires $\binom{n}{3} \cdot \lceil \log q \rceil$ many bits.

Representing the natural group action on alternating trilinear forms. The action of $\mathrm{GL}(n,q)$ on $\mathrm{ATF}(n,q)$ can be represented concretely as follows. An $A = (a_{i,j}) \in \mathrm{GL}(n,q)$ sends

$$e_i^* \wedge e_j^* \wedge e_k^* \longmapsto \sum_{1 \le r < s < t \le n} \det \begin{bmatrix} a_{i,r} & a_{i,s} & a_{i,t} \\ a_{j,r} & a_{j,s} & a_{j,t} \\ a_{k,r} & a_{k,s} & a_{k,t} \end{bmatrix} e_r^* \wedge e_s^* \wedge e_t^*.$$

Since $\{e_i^* \wedge e_j^* \wedge e_k^*, 1 \le i < j < k \le n\}$ is a linear basis for $\mathrm{ATF}(n,q)$, the action of A on a general $\phi \in \mathrm{ATF}(n,q)$ can be defined by linearly extending this action.

2.4 Algorithms of group action on alternating trilinear forms

We list the algorithms of group action on alternating trilinear forms (ATFs) as follows.

Compress and decompress alternating trilinear forms. As discussed in Section 2.3, we store $\binom{n}{3}$ field elements for each ATF defined over $\mathrm{GL}(n,q)$. To facilitate some computation, we use the function $\mathsf{depressATF}$ to convert ATF into $n \times n \times n$ tensor T over $\mathbb{F}_q$ with n^3 field elements, such that $T(i,j,k) = \phi(e_i, e_j, e_k)$ for $i,j,k \in [n]$, where e_i is the ith standard basis vector. We then use the function $\mathsf{compressATF}$ to convert the tensor back to ATF. Additionally, we use $\mathsf{decompressATFS}(\{\phi_{atf}\}, c)$ and $\mathsf{compressATFS}(\{\phi_{tensor}\}, c)$ to convert multiple ATFs and tensors, where c denotes the number of ATFs or tensors.

Group actions on alternating trilinear forms.

– The function $\mathsf{actingOnATFS}(\phi_{atf}, \{A^{col}\}, c)$ outputs c ATFs by each column matrix in $\{A^{col}\}$ independent acting on ϕ_{atf}. see Section 4.2 and Appendix B for more details about column matrices.
– The function $\mathsf{invertingOnATF}(\phi_{atf}, A)$ is used as the inverse of the column matrix A acting on an ATF ϕ_{atf}.
– The function $\mathsf{columnsMatrix}(A^{col}, B^{col})$ outputs a matrix C such that $C = AB$, where A^{col} and B^{col} be the column matrix representations of A and B respectively.
– The function $\mathsf{columnsDecomposition}(A)$ outputs the column matrix representation A^{col} of A. Note there is a small probability that the matrix cannot be decomposed, then the signning algorithm will restart with a new seed in this case [8].

2.5 Hashing

In our implementation we use SHA-3 as our hash function. The SHA-3 hash function family consists of four hash functions: SHA224, SHA256, SHA384, and SHA512. Each hash function produces an output with a different length of 224, 256, 384, and 512 bits, respectively. In our signature scheme, we use SHA256, SHA384, SHA512 as our hash function corresponding to ALTEQ instance I, III, IV respectively. All of our expanders are AES-based, the following is a brief description of them; see Section 4.3 for specific implementation details.

[8] In the reference code, if the column decomposition fails, there is a while loop for the signing algorithm until it succeeds.

Generating the challenge. The function expandChallenge is used for generating challenges from the unbalanced challenge space $\mathcal{C}_{r,K}$. See Section 3.4 for unbalanced challenges.

Sampling the ATF*.* The function expandATF is used to generate the alternating trilinear forms of the signature scheme. It maps a uniform seed seed $\in \{0,1\}^\lambda$ to $\phi \in \mathrm{ATF}(n,q)$.

Expanding the column matrix. The function expandColumns is used to generate the invertible matrices. It maps a uniform seed seed $\in \{0,1\}^\lambda$ to n columns matrices represented by a matrix A.

Expanding the seeds. The function expandSeeds maps a seed seed $\in \{0,1\}^\lambda$ to some specified number of seeds.

3 Key generation, sign, and verification procedures

In this section we present the signature scheme as proposed in [TDJ+22]. Note that here C is not necessarily a power of 2 as presented in Section 1.1, since we use expander instead of simply slicing the string. In order to enhance implementation feasibility, we have made a slight modification to the basic approach. While the basic approach presented in Section 1.1 involves generating C alternating trilinear forms (ATFs) corresponding to $C-1$ invertible matrices, our new version generates $C+1$ ATFs corresponding to C invertible matrices. It is important to note that this modification does not impact the security or signature size, but merely adjusts the index.

3.1 Parameters

The ALTEQ scheme requires the following parameters:

1. n: the vector space dimension.
2. q: the finite field order.
3. λ: the security parameter in bits.
4. r: the round number.
5. $C+1$: the number of alternating trilinear form.
6. K: the weight of the challenge[9].
7. M: the message.

3.2 Key generation

The key generation procedure is described in Algorithm 1. Here we summarise the formats of private and public keys.

Private Key. The *private key* consists of C invertible matrices $A_0, \dots, A_{C-1} \in \mathrm{GL}(n,q)$.

The private key consists of

$$C \cdot n^2 \tag{1}$$

field elements. Note that the private key can be generated by a pseudo-random generator, so we only need a random seed as a private key.

[9] K is the parameter required for the unbalanced challenge technique. As will be clear later, K denotes the number of cha_i such that $\mathsf{cha}_i \neq C$.

Public Key. The *public key* consists $C+1$ alternating trilinear forms $\phi_0, \phi_1, \dots, \phi_C \in \mathrm{ATF}(n,q)$ such that $\phi_i \circ A_i = \phi_C$ for $i \in \{0, \dots, C-1\}$.

The public key consists of

$$(C+1) \cdot \binom{n}{3} \tag{2}$$

field elements. [10]

Algorithm 1: Key generation.

```
Input: The variable number $n \in \mathbb{N}$, a prime power $q$, the alternating trilinear form number $C+1$.
Output: Public key: $C+1$ alternating trilinear forms $\phi_i \in \mathrm{ATF}(n,q)$ such that $\phi_i \cong \phi_j$ for any
        $i, j \in \{0, \dots, C\}$.
Private key: $C$ matrices $A_0, \dots, A_{C-1}$, such that $\phi_i \circ A_i = \phi_C$.
1 Randomly sample an alternating trilinear form $\phi_C : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$.
2 Randomly sample $C$ invertible matrices, $A_0, \dots, A_{C-1} \in \mathrm{GL}(n,q)$.
3 For every $i \in \{0, \dots, C-1\}$, $\phi_i \leftarrow \phi_C \circ A_i$.
4 For every $i \in \{0, \dots, C-1\}$, $A_i \leftarrow A_i^{-1}$.
5 return Public key: $\phi_0, \phi_1, \phi_2, \dots, \phi_C$. Private Key: $A_0, \dots, A_{C-1}$.
```

3.3 Signature generation and verification

The algorithms 2 and 3 describe the signature generation and verification processes of our basic scheme.

Algorithm 2: Signature generation process.

```
Input: The public key $\phi_0, \dots, \phi_C \in \mathrm{ATF}(n,q)$. The private key $A_0, \dots, A_{C-1} \in \mathrm{GL}(n,q)$. $r$, $C$, $\lambda \in \mathbb{N}$.
       Let $A_C = I$, the identity matrix. The message $M$. A hash function $H : \{0,1\}^* \to \{0,1\}^{2\lambda}$. An
       expander $\mathsf{Expand} : \{0,1\}^{2\lambda} \to \{a_i\}_{i \in \{0,\dots,r-1\}}$, where $a_i \in \{0, \dots, C\}$.
Output: The signature $S$ on $M$.
1  for $i \in \{0, \dots, r-1\}$ do
2      Randomly sample $B_i \in \mathrm{GL}(n,q)$.
3      $\psi_i \leftarrow \phi_C \circ B_i$.
4  end
5  Compute $\mathsf{cha} = H(M|\psi_0|\dots|\psi_{r-1}) \in \{0,1\}^{2\lambda}$.
6  $(b_0, \dots, b_{r-1}) \leftarrow \mathsf{Expand}(\mathsf{cha})$
7  for $i \in \{0, \dots, r-1\}$ do
8      $D_i \leftarrow A_{b_i} B_i$ ;                         // Note that $\phi_{b_i} \circ D_i = \psi_i$.
9  end
10 return $S = (\mathsf{cha}, D_0, \dots, D_{r-1})$.
```

3.4 Unbalanced challenge space

In Algorithm 2, note that when $b_i = C$, we have $D_i = B_i$, a random matrix not related to the private key. Therefore, we can use a seed in place of B_i, which saves signature size as we only need to send a seed of size λ, rather than n^2 field elements. This observation leads to unbalancing the challenge space (as discussed already in [FS86]), reducing the

[10] In the implementation, it is feasible to reduce it to C forms since ϕ_C can be stored as the seed.

Algorithm 3: Verification procedure.

```
Input: The public key $\phi_0, \dots, \phi_C \in \mathrm{ATF}(n, q)$. The signature $S = (\mathsf{cha}, D_0, \dots, D_{r-1})$, $b_i \in \{0, \dots, C\}$,
       $D_i \in \mathrm{GL}(n, q)$. The message $M$. A hash function $H : \{0,1\}^* \to \{0,1\}^{2\lambda}$. An expander
       $\mathsf{Expand} : \{0,1\}^{2\lambda} \to \{a_i\}_{i \in \{0,\dots,r-1\}}$, where $a_i \in \{0, \dots, C\}$.
Output: "Yes" if $S$ is a valid signature for $M$. "No" otherwise.
1 for $i \in \{0, \dots, r-1\}$ do
2     Compute $\psi'_i = \phi_{b_i} \circ D_i$.
3 end
4 Compute $\mathsf{cha}' = H(M|\psi'_0|\dots|\psi'_{r-1}) \in \{0,1\}^{2\lambda}$.
5 $(b'_0, \dots, b'_{r-1}) \leftarrow \mathsf{Expand}(\mathsf{cha}')$
6 if for every $i \in \{0, \dots, r-1\}$, $b_i = b'_i$ then
7     return Yes
8 else
9     return No
```

signature size. Specifically, we will sample r challenges $(b_0, \dots, b_{r-1}) \in \{0, \dots, C\}^r$ with the property that $|\{i \in [r] \mid b_i = C\}| = r - K$. We denote $\mathcal{C}_{r,K}$ as such *unbalanced challenge space*. To achieve λ bits of security, we need to choose proper parameters r, K and C, such that $\binom{r}{K} \cdot C^K \geq 2^\lambda$.

3.5 The complete scheme

We now present the key generation, signing, and verification pseudocodes for ALTEQ, as depicted in Figure 1, and 2. The hash functions employed are from the SHA3 family, and the expanders are based on AES. To improve the efficiency of the scheme in handling a long message M, $H(H(M)||\psi_0||\dots||\psi_{r-1})$ is used rather than $H(M||\psi_0||\dots||\psi_{r-1})$ For a matrix B, B^{col} indicates that B is represented as a product of column matrices.

```
KGen
1 : $\mathsf{seed_{sk}} \leftarrow_R \{0,1\}^\lambda$
2 : $\{\mathsf{seed_{sk_i}}\}_{i \in \{0,\dots,C\}} \leftarrow \mathsf{expandSeeds}(\mathsf{seed_{sk}}, C+1)$
3 : $\phi_C \leftarrow \mathsf{expandATF}(\mathsf{seed_{sk_C}})$
4 : for $i \in \{0, \dots, C-1\}$ do
5 :     $A_i^{col} \leftarrow \mathsf{expandColumns}(\mathsf{seed_{sk_i}})$
6 :     $\phi_i \leftarrow \mathsf{invertingOnATF}(\phi_C, A_i^{col})$
7 : endfor
8 : return $(\mathsf{pk} = (\phi_0, \dots, \phi_{C-1}, \mathsf{seed_{sk_C}}), \mathsf{sk} = \mathsf{seed_{sk}})$
```

Fig. 1: The pseudo-code for key generation algorithm of ALTEQ.

3.6 The formulas for public key, private key, and signature sizes

The following formulas give the public key, private key, and signature sizes in *bits*, based on the parameters n, q, r, K, and C.

$$\mathrm{PubKeySize} = C \cdot \binom{n}{3} \cdot \lceil \log_2(q) \rceil + \lambda, \tag{3}$$

$$\mathrm{PriKeySize} = \lambda \tag{4}$$

$$\mathrm{SigSize} = (r - K + 2) \cdot \lambda + K \cdot n^2 \cdot \lceil \log_2(q) \rceil. \tag{5}$$

```
Sign(sk, $M$)
1 :  $\{\mathsf{seed}_{\mathsf{sk}_i}\}_{i\in\{0,\dots,C\}} \leftarrow \mathsf{expandSeeds}(\mathsf{seed}_{\mathsf{sk}}, C+1)$
2 :  $\phi_C \leftarrow \mathsf{expandATF}(\mathsf{seed}_{\mathsf{sk}_C})$
3 :  $\mathsf{seed} \leftarrow_R \{0,1\}^{\lambda}$
4 :  $\{\mathsf{seed}_i\}_{i\in\{0,\dots,r-1\}} \leftarrow \mathsf{expandSeeds}(\mathsf{seed}, r)$
5 :  for $i \in \{0,\dots,r-1\}$ do
6 :    $B_i^{col} \leftarrow \mathsf{expandColumns}(\mathsf{seed}_i)$
7 :  endfor
8 :  $\{\psi_i\}_{i\in\{0,\dots,r-1\}} \leftarrow \mathsf{actingOnATFS}(\phi_C, \{B_i^{col}\}_{i\in\{0,\dots,r-1\}}, r)$   // $\psi_i \leftarrow \phi_C \circ B_i$ for $i \in \{0,\dots,r-1\}$
9 :  $\mathsf{cha} \in \{0,1\}^{2\lambda} \leftarrow H(H(M)||\psi_0||\dots||\psi_{r-1})$   // H is a hash function
10 : $(c_0,\dots,c_{r-1}) \leftarrow \mathsf{expandChallenge}(\mathsf{cha})$   // Generating challenge in $\mathcal{C}_{r,K}$, where $c_i \in \{0,\dots,C\}$
11 : for $i \in \{0,\dots,r-1\}$ do
12 :   if $c_i == C$ then
13 :     $\mathsf{Append}(\mathsf{seed}_i, \mathsf{Sig})$   // Putting $\mathsf{seed}_i$ into Sig
14 :   else
15 :     $A_{c_i}^{col} \leftarrow \mathsf{expandColumns}(\mathsf{seed}_{\mathsf{sk}_{c_i}})$
16 :     $D_i \leftarrow \mathsf{columnsMatrix}(A_{c_i}^{col}, B_i^{col})$   // $D_i = A_{c_i} B_i$
17 :     $D_i^{col} \leftarrow \mathsf{columnsDecomposition}(D_i)$   // If column decomposion fails, then the Sign() will restart from line 3
18 :     $\mathsf{Append}(D_i^{col}, \mathsf{Sig})$   // Putting $D_i^{col}$ into Sig
19 : endfor
20 : return $\mathsf{Sig} = (\mathsf{cha}, \{\mathsf{seed}_i\}_{i \text{ s.t. } c_i=C}, \{D_i^{col}\}_{i \text{ s.t. } c_i\neq C})$

Vf(pk, $M$, Sig)
1 :  $\phi_C \leftarrow \mathsf{expandATF}(\mathsf{seed}_{\mathsf{sk}_C})$
2 :  $(c_0,\dots,c_{r-1}) \leftarrow \mathsf{expandChallenge}(\mathsf{cha})$
3 :  for $i \in \{0,\dots,r-1\}$ do
4 :    if $c_i == C$ then
5 :      $D_i'^{col} \leftarrow \mathsf{expandColumns}(\mathsf{seed}_i)$
6 :    else
7 :      $D_i'^{col} \leftarrow D_i^{col}$
8 :  endfor
9 :  $\{\psi_i'\} \leftarrow \mathsf{actingOnATFS}(\phi_C, \{D_i'^{col}\}_{i\in\{0,\dots,r-1\}}, r)$   // $\psi_i' \leftarrow \phi_C \circ D_i'$
10 : $\mathsf{cha}' \in \{0,1\}^{2\lambda} \leftarrow H(H(M)||\psi_0'||\dots||\psi_{r-1}')$
11 : if $\mathsf{cha} == \mathsf{cha}' \& \mathsf{checkInvertibility}(\{D_i'^{col}\})$ then   // if all $D_i$ are invertible matrices then return True, otherwise return False.
12 :   return $Yes$
13 : else
14 :   return $No$
```

Fig. 2: The pseudo-code for sign and verification processes of ALTEQ.

4 Implementation details

We implement several optimizations for modular arithmetic, group actions, and seed expansion. We also provide AVX2 acceleration.

4.1 Modular arithmetic

Operating on matrices and tensors require multiple computations of a sum of products of elements over $\mathbb{F}_q$. Therefore, we will use only one single modulo i.e. $q = 2^{32} - 5$. This choice allows to use a large field without using multiprecision arithmetic. Consequently, each multiplication need to be follow immediately by a modular reduction. Regarding modular addition, multiple operation can be done before a modular reduction. As q is a Pseudo-Mersenne number [Cra92], a modular reduction is done by a shift, an addition and multiplication by a constant. To guarantee than the result stays on 32 bits, a second round of modular reduction will need to be performed.

4.2 Representing invertible matrices and their actions

An invertible matrix is represented as a product of n invertible column matrices. Here, a *column matrix* is equal to the identity matrix for each coefficient but one column. Not all invertible matrices cannot be decomposed in such product (without the use of a permutation matrix), but the number of matrices not decomposable directly in such product of column matrices is negligible.

Once in the form of the product of n column matrices, a matrix can be applied to an alternating trilinear form in a simpler and faster way: each column matrix, one after the other, can be applied directly to the alternating trilinear without passing by a costly tensor form. We include the details in Appendix B. Consequently, we obtain a reduction from $7/4 \cdot n^4$ to $1/2 \cdot n^4$ of the number of field multiplications required. This gain is especially evident in the verification process, as a majority of the cases are expanded from random seeds.

Finally, it is important to note that we can efficiently compute the matrix corresponding to the product of column matrix by performing such product itself. The product of a dense matrix by a column matrix will cost n^2 field multiplications. In this scheme, we will need to compute the product of $2n$ columns matrices. However, the first n column matrices product will cost less than n^2 fields multiplications because such product are with elements corresponding to zero and therefore does not need to be computed. The reason is that the identity matrix will still have $(n-1) \cdot (n-k)$ elements equal to zero after k products by a columns matrix. This is correct if columns are ordered as in our implementation.

4.3 Seed expansion

As we have multiple random objects to generate, we try to minimize the call to seed expander. To this end, for random matrices and random ATFs, we simply randomly generate a large number of value in $[0, 2^{32})$. The elements will discarded in the rare cases that is not falling in $[0, q)$ or if is equal to 0 in the case of element of the diagonal of a column matrix.

For generating the challenges, we need multiple values with different sizes: both for the challenge value different from C and to determine where the challenge is equal to

C. This last step corresponds to pick K elements among r elements. It is important to note that to minimize the call to seed expander, the approach will be different if $r - K$ is smaller than K. For such cases, we pick $r - K$ elements among r elements for the same result. For all those reasons, when generating the challenges, we keep in a buffer each random bit generated by the seed expander to avoid unnecessary calls.

4.4 AVX2 acceleration

To fully utilize AVX acceleration, the representations of multiple ATFs have been intertwined: on the array representing ATF, the consecutive value does not correspond to the same ATF, but rather to the value having the same index in a different ATF. Concretely, the element corresponding to $ATF_r(i, j, k)$ is not followed by $ATF_r(i, j, k + 1)$ but by $ATF_{r+1}(i, j, k)$. Consequently, when we need to compute the action of different matrices on multiple ATFs, this can be done in a vectorize manner.

Hashing function and seed expansion can also take advantage of AVX acceleration. For our symmetric needs, we borrow solutions from some previous submission to NIST PQC standardization, such as Dilithium as well as from XKCP. The Dilithium team has already proposed an efficient and dedicated versions of AES utilizing AVX acceleration. Regarding Keccak, XKCP offers a version that is fully utilizing AVX as well. While these implementations have been slightly modified to fit our scheme, they should be fully credited to the Dilithium and XKCP teams, and they have been put in two separate folders, aes and keccak, to this end.

Furthermore, while our implementation has been optimized and parameterized for AVX2, it will strongly take the advantage of AVX512 as well.

Finally, our scheme could be accelerated even further by using multithreading. As it requires a dedicated implementation, we did not investigate such option to focus principally on AVX2 acceleration.

5 Performance analysis

We provide two sets of parameters for each security level I and III. The first set is called *Balanced*, and the second set is called *ShortSig*. We also make a suggestion for the security level V.

5.1 Key and signature sizes

In Table 2, we list the parameters for the balanced-ALTEQ for security levels I, III, and V. Note that for level I, the public key+signature size is below 24KB. For level III, the public key+signature size is below 80KB. For level V, the public key+signature size is below 192KB.

Parameter set	Parameters (n, q, r, K, C)	Private key Size (Bytes)	Public key Size (Bytes)	Signature Size (Bytes)	Public key + signature Size (Bytes)
I	$(13, 2^{32} - 5, 84, 22, 7)$	16	8024	15896	23920
III	$(20, 2^{32} - 5, 201, 28, 7)$	24	31944	49000	80944
V	$(25, 2^{32} - 5, 119, 48, 8)$	32	73632	122336	195968

Table 2: Key and Signature Sizes for Balanced-ALTEQ

In Table 3, we list the parameters for the ShortSig-ALTEQ for security levels I, III, and V. Note that for level I, the public key size is below 512KB and the signature size is below 10KB. For level III, the public key size is below 1MB and the signature size is below 32KB. For level V, the public key size is below 2MB and the signature size is below 64KB.

Parameter set	Parameters (n, q, r, K, C)	Private key Size (Bytes)	Public key Size (Bytes)	Signature Size (Bytes)
I	$(13, 2^{32} - 5, 16, 14, 458)$	16	523968	9528
III	$(20, 2^{32} - 5, 39, 20, 229)$	24	1044264	32504
V	$(25, 2^{32} - 5, 67, 25, 227)$	32	2088432	63908

Table 3: Key and Signature Sizes for ShortSig-ALTEQ

5.2 Performance

We test our codes on a machine with the following configurations.

- Processor: Intel Xeon E-2288G 3.7GHz 8 cores 16MB L3 Cache HT Enabled (Max Turbo Freq. 5.0GHz, Min 4.7GHz).
- Memory: 64GB.
- Operating system: Red Hat Enterprise Linux 8.6 (Ootpa).
- Compiler: gcc version 8.5.0 20210514 (Red Hat 8.5.0-10).

Our results are as follows. The numbers in the following Tables are averages over 1000 runs. We report the averages, and the medians are quite close to the averages.

parameter set		Key gen	Sign	Verify	Sign+verify
I	cycles	329285	2310789	1836795	4147584
	time (ms)	0.093	0.629	0.496	1.125
III	cycles	2121817	25965846	24075470	50041316
	time (ms)	0.582	6.986	6.483	13.469
V	cycles	5615103	42697421	37913185	80610606
	time (ms)	1.531	11.707	10.351	22.058

Table 4: Performance of Balanced-ALTEQ.

parameter set		Key gen	Sign	Verify
I	cycles	7123223	686620	326242
	time (ms)	1.902	0.194	0.092
III	cycles	18339415	6346193	4851234
	time (ms)	5.152	1.705	1.304
V	cycles	44853623	25040313	21972672
	time (ms)	12.567	6.894	5.969

Table 5: Performance of ShortSig-ALTEQ.

6 Security reductions

In this section we discuss the EUF-CMA security of ALTEQ in the random oracle model (ROM) and the quantum random oracle model (QROM). First we introduce the assumptions on which the security of ALTEQ is based. Then we will provide insights for the security reduction and lay the security foundation of our scheme.

6.1 Assumptions

The starting point is the following algorithmic problem.

The ATFE Problem. The decision version of the alternating trilinear form equivalence (ATFE) problem is as follows. Given two alternating trilinear forms $\phi, \psi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$, decide whether there exists $A \in \mathrm{GL}(n,q)$ such that $\phi = \psi \circ A$ or not.

Two variants of the basic ATFE problem are as follows. See [TDJ$^+$22, Remark 2] a discussion of their relations with the basic ATFE problem.

The psATFE Problem. The promised search version of the alternating trilinear form equivalence problem (psATFE) is the following. Given two alternating trilinear forms $\phi, \psi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$, with the promise that $\phi \sim \psi$, output $A \in \mathrm{GL}(n,q)$ such that $\phi = \psi \circ A$.

The C-psATFE Problem. The promised search version of the alternating trilinear form equivalence problem with C-instances (m-psATFE) is the following. Given C alternating trilinear forms $\phi_1, \ldots, \phi_C : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$, with the promise that $\phi_i \sim \phi_j$ for any $i, j \in [C]$, output some $A \in \mathrm{GL}(n,q)$ and $i, j \in [C]$, $i \neq j$, such that $\phi_i = \phi_j \circ A$.

The following automorphism version of the ATFE problem is also of interest. See [TDJ$^+$22, Section 3.2] for a discussion on this.

The ATFA problem. The alternating trilinear form automorphism problem (ATFA) asks the following: given a random alternating trilinear form $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$, decide whether there exists a non-identity $A \in \mathrm{GL}(n,q)$ such that $\phi \circ A = \phi$.

The following problem models the pseudorandom group action notion introduced in [JQSY19,AFMP20]. See [TDJ$^+$22, Section 4.2] for some evidences supporting that this is a hard problem.

The C-PR-psATFE-RO Problem. The pseudorandom alternating trilinear form equivalence problem with K random instances (C-PR-psATFE-RO) asks to distinguish the following two distributions.

The random distribution: C alternating trilinear forms $\phi_0, \phi_1, \ldots, \phi_{C-1} : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$, drawn as $(\phi_0, \phi_1, \ldots, \phi_{C-1}) \leftarrow_R \mathrm{ATF}(n,q)^C$.

The pseudorandom distribution: C alternating trilinear forms $\phi_0, \phi_1, \ldots, \phi_{C-1} : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$, such that: (1) $\phi_0 \leftarrow_R \mathrm{ATF}(n,q)$, and (2) for $i \in [C-1]$, $\phi_i := \phi_0 \circ A_i$, where $A_i \in_R \mathrm{GL}(n,q)$.

6.2 Security of ALTEQ

The EUF-CMA security of ALTEQ in ROM was proved in [TDJ+22, Theorem 1] assuming the hardness of the C-psATFE problem. This follows the standard arguments for the security of the Goldreich–Micali–Wigderson protocol as a Sigma protocol, as observed by many people.

The EUF-CMA security of ALTEQ in QROM was shown in [BCD+22] via two approaches, based on the works [KLS18,LZ19,DFMS19].

The first approach is based on the perfect unique response property. To satisfy this property, it is required that a random alternating trilinear form $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$ has no non-identity automorphisms. In [BCD+22], evidences were provided supporting that this is indeed the case for $n \geq 10$. However, the security reduction here is not tight.

The second approach is based on the lossy approach as in [KLS18]. This gives a tight security reduction. For the lossy approach, two properties are needed, namely the computational unique response property and the lossy property. The former translates to the hardness of the ATFA problem, and the latter translates to the hardness of the C-PR-psATFE-RO problem. In particular, the advantage of the adversary is upper bounded by the sum of the advantages of adversaries for ATFA, C-PR-psATFE-RO, and a term that is close to $1/2^\lambda$ for our choices of n and q.

Note also that these securities are preserved after incorporating multiple keys in each round and unbalanced challenges; see [BBPS21,BCD+22].

In fact the above EUF-CMA security in the QROM model can be strengthened to the stronger sEUF-CMA security, in which the adversary will win if he/she can produce a new signature for a message that he/she already saw a signature returned by the signing oracle. See [BCD+22] for more details.

6.3 Algorithms and complexity of the ATFE problem

We've shown how ATFE and its variants support the EUF-CMA security of ALTEQ in ROM and QROM in Section 6.2. In Section 6.1, we gave pointers to the literature where the relations of variants of ATFE and the basic ATFE were discussed. So it remains to present the current status of the basic ATFE problem. We briefly show some key points here; a survey of the complexity and main algorithms for ATFE can be found in Appendix A.

First, ATFE is shown in [GQT21] to be complete for a complexity class called Tensor-Isomorphism (TI). TI was recently introduced in [GQ21b], and TI-complete problems include many isomorphism problems arising from several research communities, such as coding theory, cryptography, theoretical computer science, and computational group theory. Despite research efforts from these communities, these problems are regarded as difficult to solve in practice. This gives us confidence in the worst-case hardness of ATFE. We briefly describe them in Appendix A.1.

Second, we analyze known algorithms for ATFE from Appendix A.2 to Appendix A.7, including Gröbner basis attacks, graph-theoretic algorithms by Beullens [Beu22], quantum random walks attacks, min-rank attack as well as a recent low-rank birthday attack by Qiao [Qia23]. These attacks are accounted for in our parameter selection.

It should be noted that, because of the connections with many isomorphism problems, the algorithmic techniques for ATFE have been drawn from years of research experience of these computational areas for such problems. Also, the quantum random walk attacks are quantum adaptations of the best known classical attacks. While these yield in a polynomial factor runtime speedup in the attacks, they are not accounted for in parameter selection. The reason being that they require exponential quantum memory, which is surely a scarcer resource that runtime.

7 Parameter choices

Our parameter choices are based on the following considerations.

1. Let λ be the target security level.
2. Select the dimension n based on the direct Gröbner basis attack in Section A.2.
3. Select the field order q based on the low-rank birthday attack in Section A.5.
4. Select the round number r, the unbalanced parameter K, and the form number in each round C based on Section 6.2.

7.1 The choices of n

This is set up based on the direct Gröbner basis attack in Section A.2. We compare the three modellings of polynomial systems we are aware of in Appendix A.2. We note that to estimate the solving degrees of these systems is a major open problem. Lacking proper tools to understand them, we resort to the estimates of semi-regular systems [BFSY05]. Based on the discussions in Section A.2, we estimate that the direct Gröbner basis attack based on quadratic with inverse modellings, and the results are given in Table 6.

7.2 The choices of q

After selecting n, the choice of q is based on the low-rank birthday attack in Section A.6. This relies on the rank statistics of n.

Let $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$ be an alternating trilinear form. Let $\mathbb{P}(\mathbb{F}_q^n)$ be the projective space associated with $\mathbb{F}_q^n$, consisting of lines in $\mathbb{F}_q^n$. That is, for $v \in \mathbb{F}_q^n$, $v \neq 0$, we let $\hat{v} := \{u \in \mathbb{F}_q^n \mid u = \alpha \cdot v, \alpha \in \mathbb{F}_q\}$. For $\hat{v} \in \mathbb{P}(\mathbb{F}_q^n)$, let $\mathrm{rk}_\phi(\hat{v})$ be the rank of the bilinear form $\phi_{\hat{v}} := \phi(v, \cdot, \cdot)$. When it is clear from the context, we may just write as $\mathrm{rk}(\hat{v})$.

Based on Theorem 1 from [Beu22], the following data are most relevant to our choice (see also Table 10, 11, and 12).

1. For $n = 13$, for a random ϕ, it is expected that $|\{\hat{v} \mid \mathrm{rk}_\phi(\hat{v}) = 8\}| \approx q^6$. It is also expected that $1/q^3$-fraction of ϕ has $\hat{v}$ such that $\mathrm{rk}_\phi(\hat{v}) = 6$.
2. For $n = 20$, for a random ϕ, it is expected that $|\{\hat{v} \mid \mathrm{rk}_\phi(\hat{v}) = 14\}| \approx q^9$. It is also expected that $1/q^2$-fraction of ϕ has $\hat{v}$ such that $\mathrm{rk}_\phi(\hat{v}) = 12$.
3. For $n = 25$, for a random ϕ, it is expected that $|\{\hat{v} \mid \mathrm{rk}_\phi(\hat{v}) = 18\}| \approx q^9$. It is also expected that $1/q^4$-fraction of ϕ has $\hat{v}$ such that $\mathrm{rk}_\phi(\hat{v}) = 16$.

The low-rank birthday algorithm in [Qia23] described in Section A.6 yield the following. Let $\mathsf{minrank\text{-}cost}(n, k, r)$ denote the min-rank cost for sampling a rank-r matrix from the linear span of k $n \times n$ matrices.

1. For $n = 13$, an algorithm with $O(q^3 \cdot \mathsf{minrank\text{-}cost}(13, 7, 8) \cdot 13^6)$ arithmetic operations.
2. For $n = 20$, an algorithm with $O(q^{4.5} \cdot \mathsf{minrank\text{-}cost}(20, 11, 14) \cdot 20^6)$ arithmetic operations.
3. For $n = 25$, an algorithm with $O(q^{4.5} \cdot \mathsf{minrank\text{-}cost}(25, 16, 18) \cdot 25^6)$ arithmetic operations.

We use the algorithm[11] from [BBC$^+$20] to estimate the min-rank cost as follows.

1. $\mathsf{minrank\text{-}cost}(13, 7, 8) \approx 2^{32}$.
2. $\mathsf{minrank\text{-}cost}(20, 11, 12) \approx 2^{57}$. Here we use the parameter $b = 4$ as in [BBC$^+$20].
3. $\mathsf{minrank\text{-}cost}(25, 16, 18) \approx 2^{80}$. Here we use the parameter $b = 7$ as in [BBC$^+$20].

We now summarise the arithmetic complexities for the direct Göbner basis attack and the low-rank birthday attack for $n = 13$, $n = 20$, and $n = 25$ with q being a 32-bit prime in Table 6.

	Quadratic with inverse GB, arithmetic	low-rank birthday, arithmetic
$n = 13, q = 2^{32} - 5$	$\approx 2^{143}$	$\approx 2^{150}$
$n = 20, q = 2^{32} - 5$	$\approx 2^{219}$	$\approx 2^{227}$
$n = 25, q = 2^{32} - 5$	$\approx 2^{276}$	$\approx 2^{252}$

Table 6: Arithmetic complexities of the two attacks.

The above discussions are for numbers of arithmetic operations. These already suffice for levels I and III. To translate to bit complexities, we may assume that each arithmetic operation involves $\lceil \log(q) \rceil$ bit operations[12].

7.3 The choices of C, r, and K

We use the unbalanced challenge technique as in Section 3.4. This relies on three parameters, the round number r, the unbalanced parameter K, and the form number in each round C. To achieve the λ bit security, we require that $\binom{r}{K} \cdot C^K \geq 2^\lambda$. Table 7 illustrates the bit securities of our choices of r, K, and C in Section 5.

parameter set	r	K	C	security level of ALTEQ (bit)
I	84	22	7	128.1
	16	14	458	130.6
III	201	28	7	192.0
	39	20	229	192.7
V	119	48	8	256.0
	67	25	227	256.2

Table 7: The bit security of ALTEQ for the choices of C, r and K used in Section 5.

Acknowledgement. We would like to thank Antoine Joux, Magali Bardet, Zhili Chen, Tuong Nguyen, Ward Beullens, Monika Trimoska, and Lars Ran for their help with various aspects of this project. We would also like to thank Yuto Nakano, Kazuhide Fukushima, Gilles Van Assche and the rest of KECCAK team for useful discussions concerning implementation aspects. We thank Jean-Claude Bajard for his idea to use Takagi's algorithm for inverting efficiently modulo a pseudo Mersenne.

[11] We compared the estimates below with the estimates based on the analysis of the Kipnis–Shamir system [KS99] in [VBC$^+$19], and found that the ones from [BBC$^+$20] are lower.

[12] In the case of multiplications, the bit complexity could be $O(\log^2(q))$. Here we take $\log(q)$ as a conservative estimation.

References

AFMP20. Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 411–439. Springer, 2020.

Amb07. Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007.

AS05. Manindra Agrawal and Nitin Saxena. Automorphisms of finite rings and applications to complexity of problems. In *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005, Proceedings*, pages 1–17, 2005.

AS06. Manindra Agrawal and Nitin Saxena. Equivalence of f-algebras and cubic forms. In *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, pages 115–126, 2006.

Bab16. László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697, 2016.

BBC$^+$20. Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray A. Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier A. Verbel. Improvements of algebraic attacks for solving the rank decoding and minrank problems. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I*, volume 12491 of *Lecture Notes in Computer Science*, pages 507–536. Springer, 2020.

BBPS21. Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. Less-fm: fine-tuning signatures from the code equivalence problem. In *Post-Quantum Cryptography: 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20–22, 2021, Proceedings 12*, pages 23–43. Springer, 2021.

BCD$^+$22. Markus Bläser, Zhili Chen, Dung Hoang Duong, Tuong Ngoc Nguyen, Thomas Plantard, Youming Qiao, Willy Susilo, and Gang Tang. On digital signatures based on isomorphism problems: QROM security, ring signatures, and implementations. *IACR Cryptol. ePrint Arch.*, page 1184, 2022.

BCH$^+$23. Ward Beullens, Ming-Shing Chen, Shih-Hao Hung, Matthias J. Kannwischer, Bo-Yuan Peng, Cheng-Jhih Shih, and Bo-Yin Yang. Oil and vinegar: Modern parameters and implementations. *IACR Cryptol. ePrint Arch.*, page 59, 2023.

BCP97. Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

Beu22. Ward Beullens. Graph-theoretic algorithms for the alternating trilinear form equivalence problem. *IACR Cryptol. ePrint Arch.*, page 1528, 2022.

BFFP11. Charles Bouillaguet, Jean-Charles Faugère, Pierre-Alain Fouque, and Ludovic Perret. Practical cryptanalysis of the identification scheme based on the isomorphism of polynomial with one secret problem. In *International Workshop on Public Key Cryptography*, pages 473–493. Springer, 2011.

BFP15. Jérémy Berthomieu, Jean-Charles Faugère, and Ludovic Perret. Polynomial-time algorithms for quadratic isomorphism of polynomials: The regular case. *J. Complexity*, 31(4):590–616, 2015.

BFSY05. Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Bo-Yin Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *Proc. of MEGA*, volume 5, 2005.

BFV13. Charles Bouillaguet, Pierre-Alain Fouque, and Amandine Véber. Graph-theoretic algorithms for the "isomorphism of polynomials" problem. In *Advances in Cryptology - EUROCRYPT 2013*, pages 211–227, 2013.

BHT98. Gilles Brassard, Peter Hoyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In Claudio L. Lucchesi and Arnaldo V. Moura, editors, *LATIN'98: Theoretical Informatics*, pages 163–169, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

BKP20. Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 464–492. Springer, 2020.

BMPS20. Jean-François Biasse, Giacomo Micheli, Edoardo Persichetti, and Paolo Santini. Less is more: code-based signatures without syndromes. In *Progress in Cryptology-AFRICACRYPT 2020: 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20–22, 2020, Proceedings 12*, pages 45–65. Springer, 2020.

CNP$^+$22. Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. Take your meds: Digital signatures from matrix code equivalence. *Cryptology ePrint Archive*, 2022.

Cra92. R. E. Crandall. Method and apparatus for public key exchange in a cryptographic system, 1992. *U.S. Patent number 5159632.*

DFMS19. *Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the fiat-shamir transformation in the quantum random-oracle model. In Alexandra Boldyreva and Daniele Micciancio, editors,* Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II, *volume 11693 of* Lecture Notes in Computer Science*, pages 356–383. Springer, 2019.*

FP06. *Jean-Charles Faugère and Ludovic Perret. Polynomial equivalence problems: Algorithmic and theoretical aspects. In* Advances in Cryptology - EUROCRYPT 2006*, pages 30–47, 2006.*

FS86. *Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In* Advances in Cryptology – CRYPTO 1986*, pages 186–194, 1986.*

GMW91. *Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems.* J. ACM*, 38(3):691–729, 1991.*

GQ21a. *Joshua A. Grochow and Youming Qiao. On p-group isomorphism: search-to-decision, counting-to-decision, and nilpotency class reductions via tensors. In* 36th Computational Complexity Conference*, volume 200 of* LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 16, 38. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2021.*

GQ21b. *Joshua A. Grochow and Youming Qiao. On the complexity of isomorphism problems for tensors, groups, and polynomials I: tensor isomorphism-completeness. In James R. Lee, editor,* 12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of* LIPIcs*, pages 31:1–31:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.*

GQT21. *Joshua A. Grochow, Youming Qiao, and Gang Tang. Average-case algorithms for testing isomorphism of polynomials, algebras, and multilinear forms. In* The 38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021*, page to appear, 2021. arXiv:2012.01085.*

Gro96. *Lov K Grover. A fast quantum mechanical algorithm for database search. In* Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.*

HMR$^+$10. *Sean Hallgren, Cristopher Moore, Martin Rötteler, Alexander Russell, and Pranab Sen. Limitations of quantum coset states for graph isomorphism.* J. ACM*, 57(6):34:1–34:33, November 2010.*

Jou23. *Antoine Joux. Mpc in the head for isomorphisms and group actions. Cryptology ePrint Archive, Paper 2023/664, 2023.* `https://eprint.iacr.org/2023/664`*.*

JQSY19. *Zhengfeng Ji, Youming Qiao, Fang Song, and Aaram Yun. General linear group action on tensors: A candidate for post-quantum cryptography. In Dennis Hofheinz and Alon Rosen, editors,* Theory of Cryptography - 17th International Conference, TCC 2019*, volume 11891, pages 251–281. Springer, 2019.*

KLS18. *Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In* Advances in Cryptology – EUROCRYPT 2018*, pages 552–586. Springer, 2018.*

KS99. *Aviad Kipnis and Adi Shamir. Cryptanalysis of the hfe public key cryptosystem by relinearization. In* Annual International Cryptology Conference*, pages 19–30. Springer, 1999.*

LQ17. *Yinan Li and Youming Qiao. Linear algebraic analogues of the graph isomorphism problem and the Erdős–Rényi model. In Chris Umans, editor,* 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 463–474. IEEE Computer Society, 2017.*

LZ19. *Qipeng Liu and Mark Zhandry. Revisiting post-quantum fiat-shamir. In Alexandra Boldyreva and Daniele Micciancio, editors,* Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of* Lecture Notes in Computer Science*, pages 326–355. Springer, 2019.*

MNRS07. *Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha. Search via quantum walk. In* Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 575–584, 2007.*

MNRS12. *Frédéric Magniez, Ashwin Nayak, Peter C Richter, and Miklos Santha. On the hitting times of quantum versus random walks.* Algorithmica*, 63:91–116, 2012.*

Pat96. *Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In* Advances in Cryptology – EUROCRYPT 1996*, pages 33–48, 1996.*

Qia23. *Youming Qiao. A new heuristic algorithm for alternating trilinear form equivalence, 2023. Work in progress.*

RRST23. *Lars Ran, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. Algebraic attack on the alternating trilinear form equivalence problem, 2023. presented at CBCrypto'23.*

sCDK$^+$21. *Ming shing Chen, Jintai Ding, Matthias Kannwischer, Jacques Patarin, Albrecht Petzoldt, Dieter Schmidt, and Bo-Yin Yang. Rainbow signature: One of the three nist post-quantum signature finalists.* `https://www.pqcrainbow.org/`*, 2021.*

Sze04. *Mario Szegedy. Spectra of quantized walks and a $\sqrt{\delta\epsilon}$-rule.* arXiv preprint quant-ph/0401053*, 2004.*

Tan09. *Seiichiro Tani. Claw finding algorithms using quantum walk.* Theoretical Computer Science*, 410(50):5285–5297, 2009.*

TDJ+22. *Gang Tang, Dung Hoang Duong, Antoine Joux, Thomas Plantard, Youming Qiao, and Willy Susilo. Practical post-quantum signature schemes from isomorphism problems of trilinear forms. In Orr Dunkelman and Stefan Dziembowski, editors,* Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part III*, volume 13277 of* Lecture Notes in Computer Science*, pages 582–612. Springer, 2022.*

VBC+19. *Javier Verbel, John Baena, Daniel Cabarcas, Ray Perlner, and Daniel Smith-Tone. On the complexity of "superdetermined" minrank instances. In* Post-Quantum Cryptography: 10th International Conference, PQCrypto 2019, Chongqing, China, May 8–10, 2019 Revised Selected Papers 10*, pages 167–186. Springer, 2019.*

YC04. *Bo-Yin Yang and Jiun-Ming Chen. All in the XL family: Theory and practice. In Choonsik Park and Seongtaek Chee, editors,* Information Security and Cryptology - ICISC 2004, 7th International Conference, Seoul, Korea, December 2-3, 2004, Revised Selected Papers*, volume 3506 of* Lecture Notes in Computer Science*, pages 67–86. Springer, 2004.*

A Complexity and algorithms for ATFE

In this appendix, we provide an overview of the complexity and algorithms for ATFE.

A.1 ATFE from the computational complexity viewpoint

The Tensor Isomorphism-complete class (TI) introduced in [GQ21b] captures many isomorphism problems arising from multivariate cryptography, machine learning, quantum information, coding theory, and computer algebra. In [GQT21], ATFE was proved to be TI-complete. Among those TI-complete problems, the following algorithmic problems are of particular relevance.

Definition 1. *The 3-tensor isomorphism problem (3TI) is the following.*

Input *Two 3-way arrays $D = (d_{i,j,k}), E = (e_{i,j,k})$, where $d_{i,j,k}, e_{i,j,k} \in \mathbb{F}_q$ and $i, j, k \in [n]$.*
Output *"Yes" if there exist $A = (a_{i,r}), B = (b_{j,s}), C = (c_{k,t}) \in \mathrm{GL}(n, q)$, such that $D = (A, B, C) \star E$, where $(A, B, C) \star E := F = (f_{i,j,k})$, $f_{i,j,k} = \sum_{r,s,t\in[n]} a_{i,r} b_{j,s} c_{k,t} e_{r,s,t}$. "No" otherwise.*

The 3TI is just the matrix code equivalence problem, on which the scheme MEDS is based on [CNP+22].

Definition 2. *The cubic form isomorphism problem (CFI) is the following.*

Input *Two cubic forms (homogeneous degree-3 polynomials) $f, g \in \mathbb{F}_q[x_1, \dots, x_n]$.*
Output *"Yes" if there exists $A = (a_{i,j}) \in \mathrm{GL}(n, q)$, such that $f = A \star g$, where the action of A on g is by sending x_i to $\sum_{j\in[n]} a_{i,j} x_j$. "No" otherwise.*

CFI has been studied in multivariate cryptography [BFFP11] and theoretical computer science [AS05,AS06].

Definition 3. *The quadratic form map isomorphism problem (QFMI) is the following.*

Input *Two tuples of quadratic forms $\mathbf{f} = (f_1, \dots, f_m)$, $\mathbf{g} = (g_1, \dots, g_m)$, where $f_i, g_j \in \mathbb{F}_q[x_1, \dots, x_n]$ are quadratic forms (homogeneous degree-2 polynomials).*
Output *"Yes" if there exist $A = (a_{i,j}) \in \mathrm{GL}(n, q)$, $B = (b_{i,j}) \in \mathrm{GL}(m, q)$, such that $\forall i \in [m]$, $f'_i = A \star g_i$, where $f'_i = \sum_{j\in[m]} b_{i,j} f_j$, and the action of A on g_i is by sending x_i to $\sum_{j\in[n]} a_{i,j} x_j$. "No" otherwise.*

QFMI has been studied in multivariate cryptography. It was first raised by Patarin [Pat96] and has been studied in several works including [FP06,BFV13,BFP15].

The polynomial-time equivalence between ATFE, 3TI, CFI and QFMI suggest that ATFE is a difficult algorithmic problem, at least from the worst-case analysis viewpoint.

Finally we mention the following linear code monomial equivalence problem.

Definition 4. *The linear code monomial equivalence problem (LCME) is the following.*

Input *Two $d \times n$ matrices over $\mathbb{F}_q$, $D, E \in \mathrm{M}(d \times n, q)$*
Output *"Yes" if there exist $A \in \mathrm{GL}(d, q)$, B an $n \times n$ monomial matrix*[13] *over $\mathbb{F}_q$, such that $D = AEB$. "No" otherwise.*

LCME is used as the security basis of the LESS scheme [BMPS20]. In [GQ21a], it was shown that LCME reduces to 3TI, and therefore to ATFE too.

A.2 The direct Gröbner basis attack

Let $\phi, \psi \in \mathrm{ATF}(n, q)$ be two alternating trilinear forms. We wish to decide if there exists $A \in \mathrm{GL}(n, q)$ such that $\phi = \psi \circ A$. The Gröbner basis attack is the following. First, formulate a polynomial system whose solutions are isomorphisms from ϕ to ψ. Second, use the polynomial solvers, such as Göbner basis and XL, to solve such systems.

Two ways of formulating as polynomial systems are as follows. Both depend on the following data from ϕ and ψ.

From $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$ be an alternating trilinear form. Then construct a matrix tuple $\mathtt{A} = (A_1, \ldots, A_n) \in \mathrm{M}(n, q)^n$, where $A_k(i, j) = \phi(e_i, e_j, e_k)$. Recall that e_i is the ith standard basis vector.

Similarly, from $\psi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$, we construct $\mathtt{B} = (B_1, \ldots, B_n) \in \mathrm{M}(n, q)^n$.

The direct cubic modelling. The following modelling is straightforward. Let $X = (x_{i,j})_{i,j \in [n]}$ be an $n \times n$ variable matrices. Set up the following equations.

1. For $i \in [n]$, set $\sum_{j \in [n]} x_{i,j} \cdot X^t A_i X = B_i$.

Note that by alternating, the above setup uses n^2 variables to set up $\binom{n}{3}$ inhomogeneous equations. Also note that here we do not need to impose that X is invertible, because ϕ and ψ are non-degenerate[14] with high probability.

The quadratic with inverse modelling. This is the formulation studied in [TDJ$^+$22], which traced back to [BFFP11] for cubic form equivalence.

Let $X = (x_{i,j})_{i,j \in [n]}$ and $Y = (y_{i,j})_{i,j \in [n]}$ be two $n \times n$ variable matrices. Set up the following equations.

1. Set $XY = I_n$ and $YX = I_n$. This imposes that X and Y are inverses to each other.
2. For $i \in [n]$, set $X^t A_i X = \sum_{j \in [n]} y_{i,j} B_j$, and $Y^t B_i Y = \sum_{j \in [n]} x_{i,j} A_j$.

The above setup uses $2n^2$ variables to set up $2n^2 + 2 \cdot n \cdot \binom{n}{2} = 2n(\binom{n}{2} + n)$ inhomogeneous quadratic equations.

[13] That is, a matrix with each row and each column containing exactly one non-zero entry.
[14] ϕ is degenerate if there exists a non-zero vector $u \in \mathbb{F}_q^n$ such that for every $v, w \in \mathbb{F}_q^n$, $\phi(u, v, w) = 0$.

The quadratic dual modelling. This formulation is due to [RRST23]. Let $X = (x_{i,j})_{i,j\in[n]}$ be an $n \times n$ variable matrix. Let y be a variable.

Let $\ell = \binom{n}{2} - n$, and let $C_1, \dots, C_\ell$ be a basis of the linear space $\{D \in \Lambda(n,q) \mid \mathrm{Tr}(B_i D^t) = 0\}$, where Tr denotes taking the trace of a matrix.

Set up the following equations.

1. For $i \in [n]$, $j \in [\ell]$, $\mathrm{Tr}(X^t A_i X D^t) = 0$.
2. Let the $(1,2)$ entry of $X^t A_1 X$ be q, which is a homogeneous quadratic polynomial in $x_{i,j}$. Set $q \cdot y = 1$.

The above setup uses $n^2 + 1$ variables to set up $(\binom{n}{2} - n) \cdot n + 1$ equations. Among them, $(\binom{n}{2} - n) \cdot n$ are homogeneous quadratic polynomials in n^2 variables. The extra cubic equation, $q \cdot y = 1$, is introduced to prevent some undesirable solutions such as rank-1 matrices[15].

Practical evaluations of the three modellings. We carried out experiments for all the above methods on Magma [BCP97].

All work for $n = 5$ on a laptop[16].

Modelling	Direct cubic	Quadratic with inverse	Quadratic dual
Time	< 0.01s	≈ 35s	≈ 11s
Step	4	15	13
Max degree	7	7	7
Memory	900MB	800 to 900MB	800 to 900 MB

Table 8: Performance of the three modellings for $n = 5$.

For $n = 6$, we put the experiments on a server[17].

Modelling	Direct cubic	Quadratic with inverse	Quadratic dual
Time	≈ 300s	between 79000s and 90000s	Could not finish after three weeks
Step	21	48	5 (stuck at)
Max degree	7	7	7 (stuck at)
Memory	4.2GB	167GB	170GB

Table 9: Performance of the three modellings for $n = 6$.

For $n = 7$, the direct cubic modelling failed after taking more than 300GB memory.

We computed the Hilbert series for the homogeneous parts of the three modellings, and they do not resemble generic polynomial systems with the same variable and equation numbers. To estimate the solving degrees and to investigate into these modellings is an open problem.

[15] The authors of [RRST23] did not include this cubic equation. Without this cubic equation, all rank-1 matrices and some rank-2 matrices are in the solution space, which is problematic for the XL method. Furthermore, the semi-regularity assumption made in [RRST23] does not hold, as some syzygies do appear, which need to be taken into account for the regularity calculations. Given these considerations as well as the practical performances in Table 9, we chose not to adopt their estimations presented at CBCrypto when preparing this document. We thank the authors of [RRST23] for communicating their discoveries to us.

[16] MacBook Pro, Apple M1 Pro chip, 32 GB memory.

[17] 2x AMD EPYC 7532 2.40GHz 32 cores 256M L3 Cache (Max Turbo Freq. 3.33GHz), 1024GB 3200MHz ECC DDR4-RAM (Eight Channel).

Estimations based on semi-regular assumptions. We therefore adopt the following approach as a guide. We are aware that these systems are not homogeneous nor semi-regular, so this approach should not be applicable. But we resort to it due to the lack of appropriate tools at the moment.

First, we decide to follow the (*unrealistic*) assumption that these systems behave as semi-regular systems. Second, the regularity for cubic systems is usually much larger than quadratic ones, so for the sake of conservation, we drop the direct cubic modelling, despite that its performance is better than the other two. Third, we drop the quadratic dual modelling, because its performance at $n = 6$ is much worse than the other two (see also Footnote 15).

This leaves us with the quadratic with inverse modelling. Following [YC04], we compute the regularity degrees d, use $\binom{2n^2+d}{d}$ as the Macaulay matrix sizes and $2 \cdot \binom{n}{2} + n$ as the density. Based on the formula $3 \cdot (\mathsf{Macaulay\text{-}mat\text{-}size})^2 \cdot \mathsf{density}$ as used in Rainbow [sCDK+21] and UOV [BCH+23], we have the following estimates for the number of *arithmetic operations.*

1. $n = 13$, regularity $d = 11$, $\mathsf{Macaulay\text{-}mat\text{-}size} \approx 2^{67}$, and arithmetic operations $\approx 2^{143}$.
2. $n = 20$, regularity $d = 15$, $\mathsf{Macaulay\text{-}mat\text{-}size} \approx 2^{104}$, and arithmetic operations $\approx 2^{219}$.
3. $n = 25$, regularity $d = 18$, $\mathsf{Macaulay\text{-}mat\text{-}size} \approx 2^{132}$, and arithmetic operations $\approx 2^{276}$.

A.3 The Gröbner basis attack with partial information

In the following we follow the quadratic with inverse modelling as this is more thoroughly studied.

The information of one row of X. Recall that in the quadratic with inverse modelling, there are two variable matrices X and Y. It has been observed from experiments that, if the entries on the first column of X is set to scalars, then the Gröbner basis execution runs in estimated time $O(n^{2\cdot\omega} \cdot \log(q))$ [TDJ+22, Assumption 1]. This observation goes back to [BFFP11] for cubic form equivalence. Geometrically, this means that we've guessed the image of some vector v under the matrix X.

When $n \leq 8$. When $n \leq 8$, we have $n^2 > \binom{n}{3}$, that is, $|\operatorname{GL}(n, q)| > |\operatorname{ATF}(n, q)|$ for $n \leq 8$. This allows to randomly set some entries of X while still preserving the existence of a solution. This is in the spirit of hybrid Gröbner basis algorithms and also exploit the fact that many solutions may exist, especially in small dimension. It is reported in [TDJ+22] that, this method leads to a very fast attack on $n = 7$ and permits breaking $n = 8$. However, $n = 9$ remains out of range of this improvement.

A.4 Rank statistics of random alternating trilinear forms

Let $\mathbb{P}(\mathbb{F}_q^n)$ be the projective space associated with $\mathbb{F}_q^n$, consisting of lines in $\mathbb{F}_q^n$. That is, for $v \in \mathbb{F}_q^n$, $v \neq 0$, we let $\hat{v} := \{u \in \mathbb{F}_q^n \mid u = \alpha \cdot v, \alpha \in \mathbb{F}_q\}$. For $\hat{v} \in \mathbb{P}(\mathbb{F}_q^n)$, let $\operatorname{rk}_\phi(\hat{v})$ be the rank of the bilinear form $\phi_{\hat{v}} := \phi(v, \cdot, \cdot)$. When it is clear from the context, we may just write as $\operatorname{rk}(\hat{v})$.

Let $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$ be a random alternating trilinear form. That is, each of the $\binom{n}{3}$ coefficients is uniformly randomly sampled from $\mathbb{F}_q$. For given n, q, and $r \in \mathbb{N}$, we are interested in the average number of $\hat{v}$ such that $\phi_{\hat{v}}$ is of rank r. Experimental data of such distributions for small n and q were shown in [TDJ+22], and [Beu22, Theorem 1] gave formulas for such distributions.

Theorem 1 ([Beu22, Theorem 1]). *Let $\phi \in \mathrm{ATF}(n,q)$ be an alternating trilinear form. Let $d, d_1, d_2 \in [n]$ such that $n-d$, $n-d_1$, $n-d_2$ are even numbers. Let $G(\phi,d) := \{\hat{v} \in \mathbb{P}(\mathbb{F}_q^n) \mid \mathrm{rk}_\phi(\hat{v}) = n-d\}$, and $G(\phi,d_1,d_2) := \{(\hat{v_1},\hat{v_2}) \mid v_2 \in \ker(\phi_{\hat{v}}), \mathrm{rk}_\phi(\hat{v_1}) = n-d_1, \mathrm{rk}_\phi(\hat{v}) = n-d_2\}$.*

As $q \to \infty$, the average size of $|G(\phi,d)|$ over a uniformly randomly sampled alternating trilinear form ϕ tends to $q^{n-2+(-d^2+3d)/2}$, and the average size of $|G(\phi,d_1,d_2)|$ over a uniformly randomly sampled alternating trilinear form ϕ tends to $q^{n-6+(-d_1^2-d_2^2+5(d_1+d_2))/2}$.

The following Tables 10, 11, 12 are based on Theorem 1.

rank	12	10	8	6
count	q^{12}	q^{11}	q^6	$1/q^3$

Table 10: The rank statistics for $n = 13$.

rank	18	16	14	12
count	q^{19}	q^{16}	q^9	$1/q^2$

Table 11: The rank statistics for $n = 20$.

rank	24	22	20	18	16
count	q^{24}	q^{23}	q^{18}	q^9	$1/q^4$

Table 12: The rank statistics for $n = 25$.

Weak keys. Based on Theorem 1, Beullens noted that for some n, there are weak keys [Beu22]. Take $n = 10$ as an example. The highest rank in this setting is 8. By Theorem 1, with probability $\sim 1/q$, there is a unique $\hat{v}$ of rank 4. This can then be combined with the Gröbner basis with partial information method, to give a fast algorithm, as the problem completely boils down to find this unique $\hat{v}$.

Because of this, we choose $n = 13, 20$, and 25, for which the probability of generating weak keys for a 32-bit prime q is at most $1/2^{64}$.

A.5 The low-rank collision attack

In [Beu22], Beullens designed novel algorithms for ATFE. We describe the algorithm that is most relevant to our setting, and refer the readers to other beautiful algorithms in [Beu22]. We call the following algorithm the *low-rank collision attack*.

The main idea. The main idea is as follows. Let $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$. Suppose by Theorem 1, it is expected that there are roughly q^k many $\hat{v} \in \mathbb{P}(\mathbb{F}_q^n)$, such that $\mathrm{rk}_\phi(\hat{v}) = r$.

To test isomorphism from ϕ to ψ, we can first sample $q^{k/2}$ rank-r (projective) points each from ϕ and ψ, and then find a collision, i.e. $(\hat{u},\hat{v})$ such that the isomorphism $A(\hat{u}) = \hat{v}$, via the Gröbner basis with partial information method[18].

[18] As in [Beu22], the Gröbner basis with partial information method needs to be strengthened as follows. Suppose $v \in \mathbb{F}_q^n$ satisfies that the rank of the bilinear form $\phi_{\hat{v}} : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$ is $r < n$. Then it is sufficient to guess

Therefore, the cost of the low-rank collision attack is of the following form:

$$O(q^{k/2} \cdot \textsf{samp-cost} + q^k \cdot \textsf{col-cost}).$$

The collision cost col-cost can be estimated as $O(n^6)$ from experiments. The sampling cost samp-cost refers to the cost of sampling a rank-r (projective) point.

The straightforward way to sample a rank-r point is to formulate it as a min-rank problem. We will discuss this further in Section A.6. Beullens' main novel contribution lies in the sampling step, which we call the graph walking method.

The graph associated with an alternating trilinear form. Let $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$ be an alternating trilinear form. The graph associated with ϕ is $G(\phi) = (V, E)$ where $V = \mathbb{P}(\mathbb{F}_q^n)$, and for $u, v \in \mathbb{P}(\mathbb{F}_q^n)$, $\{u, v\} \in E$ if and only if $\phi(u, v, x) = 0$ for all $x \in \mathbb{F}_q^n$. Note that $\{u, v\} \in E$ if and only if the linear form obtained by instantiating the first two arguments of ϕ to u and v is the zero linear form. Such graphs have been used in algorithms for other related isomorphism problems [BFV13].

We can then assign labels to the vertices of $G(\phi)$ as follows. It is clear that $\mathrm{rk}(\hat{v})$ is an isomorphism invariant, that is, if ϕ and ψ are equivalent, then any isomorphism sends $\hat{v}$ of $\mathrm{rk}(\hat{v}) = r$ to some $\hat{u}$ of the same label.

The graph walking method. To sample a rank-r point $\hat{v}$, Beullens uses the graph walk method. That is, suppose we start with $\hat{v}$ of a large rank (i.e. $\mathrm{rk}(\hat{v}) = n - 1$ if n is odd, and $\mathrm{rk}(\hat{v}) = n - 2$ if n is even). It is easy to compute the neighbours of $\hat{v}$ on $G(\phi)$ by computing the kernel of the matrix $\phi(v, \cdot, \cdot)$. Then the question of whether the kernel contains a low-rank vector $\hat{u}$ can be modelled as a min-rank problem with only $n - r$ matrices. We can use the estimate of min-rank by Bardet et al [BBC$^+$20]. Combining with the rank distributions, the probability of the neighbours in $\hat{v}$ having a small-rank one can be computed. This leads to a sampling procedure of low-rank vectors.

For example, when $n = 13$, we can set $r = 8$. By Table 10, we expect to get one rank-10 $\hat{u}$ after q samples. We also expect that, after getting q rank-10 vectors $\hat{u}$, there exists a rank-8 $\hat{v}$ in the neighbourhood of $\hat{u}$. Therefore, the total cost of sampling one rank-8 is q^2 times the min-rank cost in 3 matrices of size 13×13, which can be estimated as $O(n^{11})$ [BBC$^+$20].

A.6 The low-rank birthday attack

In [Qia23], a new heuristic algorithm for ATFE was proposed. The main innovation of that algorithm is to associate distinguishing isomorphism invariants to low-rank points.

The main idea. We briefly describe the idea here. Let $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$. Suppose by Theorem 1, it is expected that there are roughly q^k many $\hat{u} \in \mathbb{P}(\mathbb{F}_q^n)$, such that $\mathrm{rk}_\phi(\hat{u}) = r$.

Let us *assume* that there is an easy-to-compute, distinguishing, isomorphism invariant[19] for those rank-r $\hat{u}$.

Then the algorithm goes as follows: first sample $O(q^{k/2})$-many rank-r points for ϕ, and $O(q^{k/2})$-many rank-r points for ψ. For each point, compute this isomorphism invariant. Then by birthday paradox, there exist one point $\hat{u}$ from ϕ and one point $\hat{v}$ from ψ such

the image of v under the matrix X *up to a scalar*, as the kernel of $\phi_{\hat{v}}$ can be incorporated to provide further information.

[19] That is a function f from low-rank points to some set S, such that $f(\hat{u}) \neq f(\hat{v})$ for $\hat{u} \neq \hat{v}$, and f is unchanged by basis changes.

that their isomorphism invariants are the same. Finally, use Göbner basis with partial information for $\hat{u}$ and $\hat{v}$ to recover the desired isomorphism.

The running time of the above algorithm can then be estimated as

$$O(q^{k/2} \cdot \textsf{samp-cost} \cdot \textsf{inv-cost}).$$

We estimate the sampling cost, samp-cost, and the invariant (computation) cost, inv-cost, as follows.

The sampling step. The sampling step can be done by either the min-rank method, or the graph-walking method. The graph-walking method involves q, which is usually large, so we use the min-rank method by [BBC+20].

Suppose we wish to sample a rank-r point $\hat{v} \in \mathbb{P}(\mathbb{F}_q^n)$ for ϕ. We construct a min-rank instance as follows. For $i \in [n]$, let A_i be the alternating matrix representing the bilinear form ϕ_{e_i}, where e_i is the ith standard basis vector. Let x_i, $i \in [n]$, be formal variables, and set $A = \sum_{i\in[n]} x_i A_i$. It is expected that there are q^k-many rank-r projective points for ϕ. So for $i \in [2 \ldots k+1]$, let $x_i = \alpha_i x_1$, where $\alpha_i \in_R \mathbb{F}_q$. This gives us a min-rank instance with $n-k$ variables of size $n \times n$.

An important note is that the min-rank instance above has some structural constraints due to alternating trilinear forms. As pointed out in [Beu22], such structures should impact the min-rank algorithm from [BBC+20] adversely. Still, we use the estimates from [BBC+20] as they should serve as a lower bound. We also compare the estimates from [BBC+20] with the analysis of the Kipnis–Shamir modelling [KS99] in [VBC+19], and found the ones from [BBC+20] are lower.

Consider an (n, k, r) minrank instance, namely finding a rank-r matrix in a linear span of k $n \times n$ matrices. First, we need to compute the smallest b such that $b < r + 2$ and

$$\binom{n}{r}\binom{k+b-1}{b} - 1 \leq \sum_{i=1}^{b} (-1)^{i+1} \binom{n}{r+i}\binom{n+i-1}{i}\binom{k+b-i-1}{b-i}.$$

Based on this b, the complexity is estimated as

$$O\big(k \cdot (r+1) \cdot (\binom{n}{r} \cdot \binom{k+b-1}{b})^2\big).$$

The isomorphism invariant step. The inv-cost is the main novel step of [Qia23]. The key is to observe the following. Suppose $\hat{u} \in \mathbb{P}(\mathbb{F}_q^n)$ satisfies that $\mathrm{rk}_\phi(\hat{u}) = r$. Then $K := \ker(\phi_{\hat{u}}) \leq \mathbb{F}_q^n$ is a dimension-$(n-r)$ space, also preserved by any isomorphism. This allows us to consider the trilinear form $\hat{\phi} : K \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$, and it can be verified easily that the isomorphic type of $\hat{\phi}$ under $\mathrm{GL}(K) \times \mathrm{GL}(n, q)$ is an isomorphism invariant.

In [Qia23], experiments were carried out to show that the isomorphic type of $\hat{\phi}$ is distinguishing. Furthermore, because $u \in K$ due to alternating, to test isomorphism between two such trilinear forms can be done via the Gröbner basis with partial information method. The cost for this step can then be estimated as $O(n^6)$ based on [BFFP11].

It should be noted that, just testing isomorphism here is not enough, and canonical forms are required to serve as an isomorphism invariant. Even though to transform an isomorphism invariant algorithm to a canonical form one may not be an easy process, it is generally regarded as doable, at least from the experience from graph isomorphism [Bab16]. Therefore, we take the conservative approach, namely assuming a canonical form algorithm matching the isomorphism testing algorithm running time.

A.7 Solving ATFE and psATFE through quantum random walks

Classical naive exhaustive search algorithms for ATFE and psATFE can be quadratically sped up on a quantum computer using Grover's search [Gro96]. The speedup at best brings the asymptotic runtime down to $O(q^{n^2/2})$, since the generic orbit to search in is of size at least $\Omega\left(|GL(n,q)|\right)$. In comparison, the more sophisticated classical algorithms (in sections A.3 and A.4) are much faster. To speed up these more sophisticated classical algorithms, we deploy certain extensions of Grover's algorithm through quantum random walks on an exponentially large graph. The prospect of a quadratic speedup using simple Groverization warrants further investigation.

The first extension of Grover of relevance to us is an algorithm for collision detection due to Brassard, Høyer, and Tapp [BHT98], special to two-to-one functions. Ambanis removed these restrictions and devised improved collision detection algorithms through quantum random walks, that match lower bounds [Amb07]. Szegedy further improved these algorithms and brought them under a unified framework of quantum random walks with memory [Sze04]. We will use Szegedy's version of quantum random walks for cubic quantum speedups of classical algorithms to the decision version ATFE. Extensions of Szegedy's algorithm by Magniez, Nayak, Richter, Roland, and Santha [MNRS07,MNRS12] may be deployed to tackle the search version psATFE within the same running time. Another extension/application of Szegedy's algorithm is to claw finding, by Tani [Tan09]. The claw finding formalism is particularly convenient to phrase psATFE in and infer a cubic speed up.

In applying these quantum random walk algorithms, we will invoke generic algorithms applicable to functions on finite sets presented as an oracle. Taking into account structures special to our problem; such as the expansion of the Cayley graphs underlying our problems may lead to gains in the polynomial factors. For clarity of exposition, we refrain from exploiting such structures. Instead, we will focus on speedups to the main exponential term and suppress incremental polynomial factors.

We next describe how to deploy these quantum random walk algorithms to ATFE/psATFE resulting in polynomial improvements to the asymptotic runtime complexity. However, they come at the cost of exponential quantum memory and hence will not be considered for benchmarking.

A classical oracle from the Gröbner basis attack with partial information. Let ϕ and ψ denote the two input trilinear forms with the existence of an $A \in GL_n(\mathbb{F}_q)$ such that $\psi = \phi \circ A$ in question. Central to all our methods is a polynomial time classical algorithm to test membership in the relation set

$$R_{\phi,\psi} := \{(u,v) \in \mathbb{F}_q^n \times \mathbb{F}_q^n \mid \exists A \text{ such that } \psi = \phi \circ A \text{ and } Au = v\}.$$

If ϕ and ψ are not isomorphic, $R_{\phi,\psi}$ is empty. A pair $(u,v) \in \mathbb{F}_q^n \times \mathbb{F}_q^n$ satisfying $Au = v$ enforces n $\mathbb{F}_q$-linear constraints on A. The Gröbner basis attack with partial information in section A.3, augmented with these linear constraints can tell in polynomial time if the pair (u,v) is in $R_{\phi,\psi}$. We henceforth make the same assumptions. Further, incorporate a time out clause into the membership algorithm to make the Gröbner basis methods stop searching and declare non existence.

Solving ATFE *through quantum random walks.* We devise an algorithm for the search version ATFE through Szegedy's quantum random walk. We first paraphrase theorem 3 in [Sze04], specialized to the oracle function being the identity. Let X be a finite set and $R \subset X \times X$ a binary relation with a membership tester. For a positive real number α and a uniformly random subset $H \subset X$ of size $|X|^\alpha$, let p_α denote the probability that $R \cap (H \times H)$ is non empty. There is a quantum algorithm to differentiate between the cases $p_\alpha = 0$ and $p_\alpha \geq \epsilon$ in time $O(|X|^\alpha + 1000\sqrt{|X|^\alpha/\epsilon})$.

Invoke Szegedy's algorithm with X as $\mathbb{F}_q^n$, R as $R_{\phi,\psi}$, α as $1/3$ and uniformly sampling an $H \subset \mathbb{F}_q^n$ of size $\Theta\left(q^{n/3}\right)$. We claim that the probability gap may be taken to be $\epsilon = \Omega\left(q^{-n/3}\right)$. To prove the claim, consider two isomorphic ϕ and ψ. That is, there exists at least one $A_{\phi,\psi} \in GL(n, \mathbb{F}_q)$ such that $\psi = \phi \circ A_{\phi,\psi}$. Therefore,

$$\text{Prob}\left((R_{\phi,\psi} \cap (H \times H)) \neq \emptyset\right) \geq \text{Prob}\left((H \cap A_{\phi,\psi}(H)) \neq \emptyset\right) \geq \Omega\left(q^{-n/3}\right),$$

proving the claim. In summary, we can tell if ϕ and ψ are isomorphic in time $O(q^{n/3})$ on a quantum computer.

Solving psATFE *through quantum random walks.* This strategy also tackles the promise search version psATFE within the same running time, thanks to extensions of Szegedy's algorithm by Magniez, Nayak, Richter, Roland, and Santha [MNRS07,MNRS12]. Another extension of Szegedy's algorithm is to claw finding, by Tani [Tan09]. The claw finding formalism is convenient to phrase psATFE in and infer polynomial speed ups. Let $f : X \to Z$ and $g : Y \to Z$ be two functions between finite sets. Given oracle access to f and g, the claw finding problem is to find an $(x, y) \in X \times Y$ such that $f(x) = g(y)$, if one exists. The functions may be presented either as standard oracles or as comparison oracles. We describe the later in the quantum setting, as they suffice. A comparison oracle maps quantum states

$$|x, y, b, w\rangle \longmapsto |x, y, b \oplus [f(x) >^? g(y)], w\rangle.$$

Here, b is a bit; x and y respectively index quantum states corresponding to elements in X and Y. Fixing an ordering on Z, $[f(x) >^? g(y)]$ is a bit that is one if and only if $f(x) > g(y)$. The last register indexed by w is an ancilla for work space. For instances with X and Y of roughly the same size, Tani's algorithm finds claws on a quantum computer in time $O((|X||Y|)^{1/3})$.

To phrase psATFE as claw finding, independently draw uniformly random sets $X \subset \mathbb{F}_q^n$ and $Y \subset \mathbb{F}_q^n$, each of size $q^{n/2}$. Take $f : X \to \mathbb{F}_q^n$ as the multiplication by A map $u \longmapsto Au$ and $g : Y \to \mathbb{F}_q^n$ as the identity. The birthday paradox ensures for isomorphic ϕ and ψ that there is a solution to claw finding with constant positive probability. The algorithm for testing membership in $R_{\phi,\psi}$ from the previous subsection yields a comparison oracle. Tani's algorithm for claw finding solves psATFE in time $O(q^{n/3})$.

Low rank sampling and quantum random walks The quantum random walk method also seems to work in concert with the more sophisticated algorithms in sections A.5 and A.6. It remains to work out the details and quantify the resulting polynomial quantum speed up. Again, this will be at the cost of exponential quantum storage and will not be considered for benchmarking. For instance, if there is a rank r such that $R_{\phi,r} := \{\hat{v} \in$

$\mathbb{P}(\mathbb{F}_q^n) \mid \mathrm{rk}_\phi(\hat{v}) = r\}$ (or equivalently $R_{\psi,r} := \{\hat{v} \in \mathbb{P}(\mathbb{F}_q^n) \mid \mathrm{rk}_\psi(\hat{v}) = r\}$) is small (say $q^{\delta n}$) and it is possible to sample efficiently from $R_{\phi,r}$ (or equivalently $R_{\psi,r}$); then by setting $X = R_{\phi,r} \cup R_{\psi,r}$, we can tell if ϕ and ψ are isomorphic in time $O(q^{\delta n/3})$ on a quantum computer. Large ranks such as $r = n$ are easy to sample but correspond to large $R_{\phi,r}$ or $R_{\psi,r}$. To sample from $R_{\phi,r}$ or $R_{\psi,r}$ for small r, we may look to the rank distribution results in Buellen's low-rank collision method. The low-rank birthday attack is a more refined than low-rank collision methods in that it uses a more distinguishing invariant that the rank. We suspect that the quantum random walk method speeds up the low-rank collision method to give a $O(q^{k/3})$ quantum run time.

B Column matrix decomposition and action on trilinear forms

Definition 5. *A matrix $C \in F^{n\times n}$ is a column matrix, if it is of the form*

$$C = \begin{pmatrix} 1 \dots 0 & c_1 & 0 \dots 0 \\ \vdots \ddots \vdots & \vdots & \vdots \quad\ \ \vdots \\ 0 \dots 1 & c_{i-1} & 0 \dots 0 \\ 0 \dots 0 & c_i & 0 \dots 0 \\ 0 \dots 0 & c_{i+1} & 1 \dots 0 \\ \vdots \quad\ \ \vdots & \vdots & \vdots \ddots \vdots \\ 0 \dots 0 & c_n & 0 \dots 1 \end{pmatrix}$$

for some i.

Corollary 1. *For every invertible matrix A, there is a permutation matrix P such that such that AP is the product of $2(n-1)$ column matrices.*

Note that the diagonal matrix in the definition of $\hat{U}$ can be merged with the matrices of the factoriztion of $\hat{U}$ into column matrices.

Multiplication of alternating trilinear forms with column matrices. Let C be a column matrix with entries $c_1, \dots, c_n$ in column i. The matrix C maps an unit vector e_h to

$$Ce_h = \begin{cases} e_h & \text{if } h \neq i, \\ \displaystyle\sum_{j=1}^{n} c_j e_j & \text{otherwise.} \end{cases}$$

Let T be an alternating trilinear form, that is,

$$T = \sum_{1\le r<s<t\le n} t_{r,s,t} \cdot e_r \wedge e_s \wedge e_t.$$

We have

$$C^{\wedge 3}T = \sum_{1\le r<s<t\le n} t_{r,s,t} \cdot C(e_r) \wedge C(e_s) \wedge C(e_t).$$

C only changes e_i. e_i appears in $\binom{n-1}{2}$ of the summands. Consider each summand separately. Assume w.l.o.g. that e_i appears in the first position,

$$t_{i,r,s} \cdot e_i \wedge e_r \wedge e_s.$$

$C^{\wedge 3}$ maps this summand to

$$t_{i,r,s} \cdot \left(\sum_{i=1}^{n} c_j e_j \right) \wedge e_r \wedge e_s = \sum_{j=1}^{n} \underbrace{c_j \cdot t_{i,r,s}}_{\text{1 mult.}} \cdot e_j \wedge e_r \wedge e_s.$$

Thus, we have to compute n multiplications and $n-1$ additions (updates of the entries $e_j \wedge e_r \wedge e_s$ with $j \neq i$). Therefore, the total costs are $\binom{n-1}{2} \cdot n \leq n^3/2$ multiplications and $\binom{n-1}{2} \cdot (n-1) \leq n^3/2$ additions.

If the matrix C has only k nonzero entries in column i, then the bounds reduce to $\binom{n-1}{2} \cdot j \leq n^2 \cdot j/2$ multiplications and $\binom{n-1}{2} \cdot (j-1) \leq n^2 \cdot j/2$ additions. The LUP decomposition yields a decomposition of any invertible matrix A into $2(n-1)$ column matrix with a total of $\approx n^2$ nonzero entries. Therefore, we can implement the action of $A^{\wedge 3}$ with $n^4/2$ multiplications and $n^4/2$ additions.

In the actual implementation, we have to do a modular reduction after each application of a column matrix. Therefore, we try to minimize the number of column matrices in a decomposition of A. (Obviously, it cannot be lower than n.)

Optimal decomposition into column matrices. Let

$$A = \begin{pmatrix} 1 \dots 0 & a_1 & * \dots * \\ \vdots \ddots \vdots & \vdots & \vdots \quad \vdots \\ 0 \dots 1 & a_{i-1} & * \dots * \\ 0 \dots 0 & a_i & * \dots * \\ 0 \dots 0 & a_{i+1} & * \dots * \\ \vdots \quad \vdots & \vdots & \vdots \ddots \vdots \\ 0 \dots 0 & a_n & * \dots * \end{pmatrix}.$$

If $a_i \neq 0$, then let B be the column matrix

$$B = \begin{pmatrix} 1 \dots 0 & -a_1/a_i & 0 \dots 0 \\ \vdots \ddots \vdots & \vdots & \vdots \quad \vdots \\ 0 \dots 1 & -a_{i-1}/a_i & 0 \dots 0 \\ 0 \dots 0 & 1/a_i & 0 \dots 0 \\ 0 \dots 0 & -a_{i+1}/a_i & 1 \dots 0 \\ \vdots \quad \vdots & \vdots & \vdots \ddots \vdots \\ 0 \dots 0 & -a_n/a_i & 0 \dots 1 \end{pmatrix}.$$

Then

$$BA = \begin{pmatrix} 1 \dots 0 & 0 & * \dots * \\ \vdots \ddots \vdots & \vdots & \vdots \quad \vdots \\ 0 \dots 1 & 0 & * \dots * \\ 0 \dots 0 & 1 & * \dots * \\ 0 \dots 0 & 0 & * \dots * \\ \vdots \quad \vdots & \vdots & \vdots \ddots \vdots \\ 0 \dots 0 & 0 & * \dots * \end{pmatrix}.$$

By using induction, we can find column matrices $B_1, \dots, B_n$ with B_i having column i such that

$$B_n B_{n-1} \cdots B_1 A = I.$$

The inverse of B is

$$B^{-1} = \begin{pmatrix} 1 \dots 0 & a_1 & 0 \dots 0 \\ \vdots \ddots \vdots & \vdots & \vdots \quad\; \vdots \\ 0 \dots 1 & a_{i-1} & 0 \dots 0 \\ 0 \dots 0 & a_i & 0 \dots 0 \\ 0 \dots 0 & a_{i+1} & 1 \dots 0 \\ \vdots \quad\; \vdots & \vdots & \vdots \ddots \vdots \\ 0 \dots 0 & a_n & 0 \dots 1 \end{pmatrix}.$$

We can write

$$A = B_1^{-1} B_2^{-1} \cdots B_n^{-1}.$$

By counting dimension, it is obvious that there cannot be a shorter decomposition of A into column matrices.