From: pqc-forum@list.nist.gov on behalf of Markku-Juhani O. Saarinen <mjos.crypto@gmail.com>
Sent: Tuesday, July 18, 2023 1:14 PM
To: pqc-forum
Subject: [pqc-forum] OFFICIAL COMMENT: ALTEQ

Hi All,

I'll describe a simple signature forgery attack against ALTEQ.

An ALTEQ signature consists of three parts: (cha, seed_i, D_i). ALTEQ is built on the Fiat-Shamir transform, and the verify function checks cha==cha' where cha' is reconstructed using the public key and the message. We observe that the challenge space can be reduced to binomial(r,K) size simply by setting the "seed_i" and "D_i" parts of the signature to zeros. The attack consists of finding a matching expandChallenge() output in this reduced space.

We demonstrate the attack with a full forgery against Level I ShortSig-ALTEQ. The attack is especially easy in this case as we have r=16 and K=14.

The signed message "sm", expressed in Python as follows:

sm = (bytes.fromhex('E4E7C61518AD2CE12B20D96734B665C0E7F61286186D21B1FD4BF5BD7019BAA3') + (b'\x00' * 9496) + b'Forgery')

Passes as valid with the first public key of the ShortSig-ALTEQ level I test vectors (in file `[..]/ref_mode_lp/1/PQCsignKAT_16.rsp`, starting `pk = 9F4602C4C84A05..`) I have checked this against the reference C implementation.

The forged signature consists of a 32-bit challenge hash cha=E4E7C6.. with the rest of the signature (9496 bytes) set to zeros. This is a valid signature for 7-byte ASCII text 'Forgery' or 466F7267657279.

During verification of this signature, the input to the challenge hash cha' = H(H(M) | psi'0 | psi'1 '| ..) on line 11 in Vf function of Fig 2 of the spec) becomes:
idx [len] hex
H(M): [32] 67a14a46b32990b13d97fa4961c9baed4ba64d09b24c70e199f981d41824e70a
psi0: [1144] 83044487cf6021aaa5ae526928fd54d4468e27a3810abd02d4bb08d86257ec44..
psi1: [1144] 00000000000000000000000000000000000000000000000000000000000000000000..
psi2: [1144] 00000000000000000000000000000000000000000000000000000000000000000000..
psi3: [1144] 00000000000000000000000000000000000000000000000000000000000000000000..
psi4: [1144] 00000000000000000000000000000000000000000000000000000000000000000000..
psi5: [1144] 00000000000000000000000000000000000000000000000000000000000000000000..
psi6: [1144] 00000000000000000000000000000000000000000000000000000000000000000000..
psi7: [1144] 00000000000000000000000000000000000000000000000000000000000000000000..
psi8: [1144] 00000000000000000000000000000000000000000000000000000000000000000000..
psi9: [1144] 00000000000000000000000000000000000000000000000000000000000000000000..
psi10: [1144] 00000000000000000000000000000000000000000000000000000000000000000000..
psi11: [1144] 00000000000000000000000000000000000000000000000000000000000000000000..
psi12: [1144] 00000000000000000000000000000000000000000000000000000000000000000000..
psi13: [1144] 00000000000000000000000000000000000000000000000000000000000000000000..
psi14: [1144] 83044487cf6021aaa5ae526928fd54d4468e27a3810abd02d4bb08d86257ec44..

psi15: [1144] 00000000000000000000000000000000000000000000000000000000000000000000...

This is a collision as the two (r-K) non-zero vectors are expanded at the same locations by expandChallenge. We only had to perform binomial(16,14)=120 trials to find a match. Note that the non-zero vector 1144-byte vector 8304.. depends only on the public key.

The attack complexity is log2(binomial(r,K)) bits. One can observe the r and K parameters from Tables 2 and 3 in the specification -- the attack breaks all parameter sets much faster than indicated by the security category (at most 2^113 effort, even at Level 5.) The challenge space was apparently intended be binomia(r,K)*C^K which closely matches 2^lambda for all parameter sets.

An obvious countermeasure would be to filter the signature space against such pathological cases, but this was not done in the specification or the reference implementation.

Cheers,
-markku

Markku-Juhani O. Saarinen
--

From: pqc-forum@list.nist.gov on behalf of Markku-Juhani O. Saarinen <mjos.crypto@gmail.com>
Sent: Tuesday, July 18, 2023 2:23 PM
To: pqc-forum
Cc: Markku-Juhani O. Saarinen
Subject: [pqc-forum] Re: OFFICIAL COMMENT: ALTEQ

Hi,

Typo below: It should be obvious but the challenge hash cha is 32 bytes, not 32 bits (it's included in full).

I put my Python implementation of ALTEQ together with the forgery demonstration here:
https://github.com/mjosaarinen/alteq-py

Test vectors, of course, don't guarantee that the Python verification function is correct -- the forgeries have also been checked against the C reference code.

Cheers,
- markku

On Tuesday, July 18, 2023 at 6:13:46 PM UTC+1 Markku-Juhani O. Saarinen wrote:

> Hi All,
>
> I'll describe a simple signature forgery attack against ALTEQ.
>
> An ALTEQ signature consists of three parts: (cha, seed_i, D_i). ALTEQ is built on the Fiat-Shamir transform, and the verify function checks cha==cha' where cha' is reconstructed using the public key and the message. We observe that the challenge space can be reduced to binomial(r,K) size simply by setting the "seed_i" and "D_i" parts of the signature to zeros. The attack consists of finding a matching expandChallenge() output in this reduced space.
>
> We demonstrate the attack with a full forgery against Level I ShortSig-ALTEQ. The attack is especially easy in this case as we have r=16 and K=14.
>
> The signed message "sm", expressed in Python as follows:
>
> sm = (bytes.fromhex('E4E7C61518AD2CE12B20D96734B665C0E7F61286186D21B1FD4BF5BD7019BAA3') + (b'\x00' * 9496) + b'Forgery')
>
> Passes as valid with the first public key of the ShortSig-ALTEQ level I test vectors (in file `[..]/ref_mode_lp/1/PQCsignKAT_16.rsp`, starting `pk = 9F4602C4C84A05..`) I have checked this against the reference C implementation.
>
> The forged signature consists of a 32-bit challenge hash cha=E4E7C6.. with the rest of the signature (9496 bytes) set to zeros. This is a valid signature for 7-byte ASCII text 'Forgery' or 466F7267657279.
>
> During verification of this signature, the input to the challenge hash cha' = H(H(M) | psi'0 | psi'1 '| ..) on line 11 in Vf function of Fig 2 of the spec) becomes:
> idx [len] hex
> H(M): [32] 67a14a46b32990b13d97fa4961c9baed4ba64d09b24c70e199f981d41824e70a
> psi0: [1144] 83044487cf6021aaa5ae526928fd54d4468e27a3810abd02d4bb08d86257ec44..

From: pqc-forum@list.nist.gov on behalf of Youming Qiao <jimmyqiao86@gmail.com>
Sent: Wednesday, July 19, 2023 12:25 AM
To: pqc-forum
Cc: Markku-Juhani O. Saarinen
Subject: [pqc-forum] Re: OFFICIAL COMMENT: ALTEQ

Dear Markku, Ward, and all,

I am Youming, a member of the ALTEQ team.

Thank you very much, Markku, for spotting this issue. And thank you very much, Ward, for providing your insight.

Ward already summarised the situation well. So I am writing mostly to confirm the following:

1. Markku's attack works for the current implementation.
2. This is due to an oversight in the implementation, namely we didn't check the invertibility of matrices in the input to the verification procedure.
3. The ALTEQ specification requires these matrices to be invertible.
4. Adding such an invertibility check should invalidate this attack, and this check should not be expensive.

We are working on an update version of the code and we will provide an update at https://pqcalteq.github.io/ when we are done.

We thank Markku again for spotting this issue and providing a concrete code to implement this attack so quickly. This is impressive!

We welcome and look forward to more attention and efforts on the study of ALTEQ.

Best regards,
Youming Qiao

On Wednesday, 19 July 2023 at 4:22:59 am UTC+10 Markku-Juhani O. Saarinen wrote:

> Hi,
>
> Typo below: It should be obvious but the challenge hash cha is 32 bytes, not 32 bits (it's included in full).
>
> I put my Python implementation of ALTEQ together with the forgery demonstration here:
> https://github.com/mjosaarinen/alteq-py
>
> Test vectors, of course, don't guarantee that the Python verification function is correct -- the forgeries have also been checked against the C reference code.
>
> Cheers,
> - markku
>
> On Tuesday, July 18, 2023 at 6:13:46 PM UTC+1 Markku-Juhani O. Saarinen wrote:
>
> > Hi All,
> >
> > I'll describe a simple signature forgery attack against ALTEQ.

From: pqc-forum@list.nist.gov on behalf of Markku-Juhani O. Saarinen <mjos.crypto@gmail.com>
Sent: Saturday, September 2, 2023 9:39 AM
To: pqc-forum
Cc: Youming Qiao; Markku-Juhani O. Saarinen
Subject: [pqc-forum] Re: OFFICIAL COMMENT: ALTEQ

Thanks Youming.

Recently ALTEQ security popped up in another context -- based on that feedback, I need to note that in my view, ALTEQ should be considered broken until the team updates the specification to counter this type of attack (the specification does not contain mechanisms or checks that could be directly used as a countermeasure). The countermeasure may also affect running times, as I don't see an entirely trivial check that is also sufficient -- even though the particular "backdoor signature" used in the PoC below was quite trivial. The issue is not just with all-zero inputs or plain matrix invertibility, so something like straightforward determinant computation isn't adequate.

Of course, after that update, the security of ALTEQ signature verification function reverts to essentially "unknown" until it receives some more scrutiny.

Cheers,
- markku

On Wednesday, July 19, 2023 at 5:24:40 AM UTC+1 Youming Qiao wrote:

> Dear Markku, Ward, and all,
>
> I am Youming, a member of the ALTEQ team.
>
> Thank you very much, Markku, for spotting this issue. And thank you very much, Ward, for providing your insight.
>
> Ward already summarised the situation well. So I am writing mostly to confirm the following:
>
> 1. Markku's attack works for the current implementation.
> 2. This is due to an oversight in the implementation, namely we didn't check the invertibility of matrices in the input to the verification procedure.
> 3. The ALTEQ specification requires these matrices to be invertible.
> 4. Adding such an invertibility check should invalidate this attack, and this check should not be expensive.
>
> We are working on an update version of the code and we will provide an update at https://pqcalteq.github.io/ when we are done.
>
> We thank Markku again for spotting this issue and providing a concrete code to implement this attack so quickly. This is impressive!
>
> We welcome and look forward to more attention and efforts on the study of ALTEQ.
>
> Best regards,
> Youming Qiao
>
> On Wednesday, 19 July 2023 at 4:22:59 am UTC+10 Markku-Juhani O. Saarinen wrote:

From: pqc-forum@list.nist.gov on behalf of Youming Qiao <jimmyqiao86@gmail.com>
Sent: Sunday, September 3, 2023 8:12 PM
To: pqc-forum
Cc: Markku-Juhani O. Saarinen
Subject: [pqc-forum] Re: OFFICIAL COMMENT: ALTEQ

Dear Markku-Juhani,

Thank you again for your close examination of ALTEQ. We took your attack seriously, and we had an updated version about three weeks ago.

We delayed the release, because we wished to examine other possible attacks following your approach more carefully, and took this opportunity to speedup the code further.

We aim to release a new version, hopefully sometime this week, and if not, definitely next week. Thank you for your patience.

Best regards,
Youming

On Saturday, 2 September 2023 at 11:39:23 pm UTC+10 Markku-Juhani O. Saarinen wrote:

> Thanks Youming.
>
> Recently ALTEQ security popped up in another context -- based on that feedback, I need to note that in my view, ALTEQ should be considered broken until the team updates the specification to counter this type of attack (the specification does not contain mechanisms or checks that could be directly used as a countermeasure). The countermeasure may also affect running times, as I don't see an entirely trivial check that is also sufficient -- even though the particular "backdoor signature" used in the PoC below was quite trivial. The issue is not just with all-zero inputs or plain matrix invertibility, so something like straightforward determinant computation isn't adequate.
>
> Of course, after that update, the security of ALTEQ signature verification function reverts to essentially "unknown" until it receives some more scrutiny.
>
> Cheers,
> - markku
>
> On Wednesday, July 19, 2023 at 5:24:40 AM UTC+1 Youming Qiao wrote:
>
> > Dear Markku, Ward, and all,
> >
> > I am Youming, a member of the ALTEQ team.
> >
> > Thank you very much, Markku, for spotting this issue. And thank you very much, Ward, for providing your insight.
> >
> > Ward already summarised the situation well. So I am writing mostly to confirm the following:
> >
> > 1. Markku's attack works for the current implementation.
> > 2. This is due to an oversight in the implementation, namely we didn't check the invertibility of matrices in the input to the verification procedure.
> > 3. The ALTEQ specification requires these matrices to be invertible.

From: pqc-forum@list.nist.gov on behalf of Youming Qiao <jimmyqiao86@gmail.com>
Sent: Friday, September 15, 2023 9:08 AM
To: pqc-forum
Cc: Markku-Juhani O. Saarinen
Subject: [pqc-forum] Re: OFFICIAL COMMENT: ALTEQ

Dear Markku-Juhani, dear colleagues,

We just updated the ALTEQ website (https://pqcalteq.github.io/) with a new version (v1.1). The main updates are as follows.

1. We added the invertibility check of the matrices in the verification step, so to avoid from Markku-Juhani's attack.
   In a follow-up post, we will explain our understanding of this attack, and why we believe invertibility check is enough.
2. We sped up the code further.
   Briefly speaking, for balanced parameters, we achieve speed up ~2x speed up for verification. For short signature parameters, we achieve ~4x speed up for verification. The signing time is slightly (between 6% to 25 %) faster. The key gen time is slightly (between 8% to 10%) slower for some parameter sets, and faster for other parameter sets.
   While the addition of the invertibility check was supposed to slow down the verification, we achieved a speed-up at last due to a revamped implementation (but still generic) for AVX2.

We thank Markku-Juhani for his nice attack, and Ward for his timely suggestion on understanding this attack.

We welcome more scrutiny and examinations of ALTEQ.

Best regards,
Youming (on behalf of the ALTEQ team)

On Monday, 4 September 2023 at 10:12:00 am UTC+10 Youming Qiao wrote:

> Dear Markku-Juhani,
>
> Thank you again for your close examination of ALTEQ. We took your attack seriously, and we had an updated version about three weeks ago.
>
> We delayed the release, because we wished to examine other possible attacks following your approach more carefully, and took this opportunity to speedup the code further.
>
> We aim to release a new version, hopefully sometime this week, and if not, definitely next week. Thank you for your patience.
>
> Best regards,
> Youming
>
> On Saturday, 2 September 2023 at 11:39:23 pm UTC+10 Markku-Juhani O. Saarinen wrote:
>
> > Thanks Youming.

From: pqc-forum@list.nist.gov on behalf of Youming Qiao <jimmyqiao86@gmail.com>
Sent: Friday, September 15, 2023 9:43 AM
To: pqc-forum
Cc: Markku-Juhani O. Saarinen
Subject: [pqc-forum] Re: OFFICIAL COMMENT: ALTEQ

Dear all,

In the following we explain our understanding of Markku-Juhani's attack.

=========
Synopsis of the attack.

Part 1. We discuss simplified versions of ALTEQ and the attack, as we believe that they highlight the key to Markku-Juhani's attack.

The public key of ALTEQ consists of a set of trilinear forms (phi_0, ..., phi_C) in the same orbit.

The signature of ALTEQ consists of a sequence of challenges vc=(c_0, c_1, ..., c_{r-1}), where each c_i is in {0, 1, ..., C}, and a sequence of matrices (D_0, D_1, ..., D_{r-1}).

(In ALTEQ the challenge value C is special: if c_i=C then D_i will be expanded from a seed. In this part we omit this; we will come to this one in part 2.)

Let M be the message to be signed. Let psi_i be the result of transforming phi_{c_i} by D_i. This sequence of challenges is obtained from the hash value H(H(M)|psi_0|...|psi_{r-1}).

(In the signing procedure psi_i are obtained first and then based on the hash values, D_i are computed with the help of the private key. Here we are describing the viewpoint of the verifier.)

Now Markku-Juhani's forgery (in this simplified setting) goes as follows. Let rho be the all-zero trilinear form, and A be the all-zero matrix. Compute vc=H(H(M)|rho|...|rho). Send (vc, (A, ..., A)) as the signature.

The verification will pass, because it first computes psi_i=phi_{c_i}∘A=rho, and then computes H(H(M)|rho|...|rho), which is equal to vc.

The main reason can be understood as follows: if a matrix A is not invertible, then it is possible that phi_i∘A=phi_j∘A for i, j different. In the extreme case A=0, phi_i∘A=phi_j∘A for any i, j. From the Sigma protocol viewpoint, this means that the challenge does not matter for the all-zero trilinear form as the commitment, as there is a universal reply A (the all-zero matrix) that works for any challenge.

Therefore, we believe that adding the invertibility check should invalidate this attack (and the slightly more complicated ones such as A is not zero but of rank 1).

Part 2. Further details regarding Markku-Juhani's attack.

Markku-Juhani's attack utilises the above but there is more. As mentioned in part 1, for c_i=C, D_i will be expanded from a seed. Therefore for c_i=C, the matrix expanded from a seed will be invertible, so the idea in part 1 (using all-zero matrices) does not work for this case. Markku-Juhani's clever new idea here is to insert matrices spanned from a fixed

seed at some positions. He then uses collision to identify a sequence of challenges where those positions are indeed of value C. Despite some probability of failure, this works out in practice for some parameter sets.

It is then natural to wonder if this idea would still work with the invertibility check added. Note that the difference is that it is not enough for the "collision" step to ensure that the chosen positions are of value C. It also needs to ensure that the other positions are exactly the given ones, as there is no freedom such as "any value in other positions would work". Theoretically, it can be shown that this design is EUF-CMA (assuming the hardness of ATFE and hash functions). Of course, in practice there might be attacks exploiting other types of loopholes.
=========

Best regards,
Youming (on behalf of the ALTEQ team)

On Friday, 15 September 2023 at 11:07:43 pm UTC+10 Youming Qiao wrote:

> Dear Markku-Juhani, dear colleagues,
>
> We just updated the ALTEQ website (https://pqcalteq.github.io/) with a new version (v1.1). The main updates are as follows.
>
> 1. We added the invertibility check of the matrices in the verification step, so to avoid from Markku-Juhani's attack.
>   In a follow-up post, we will explain our understanding of this attack, and why we believe invertibility check is enough.
> 2. We sped up the code further.
>   Briefly speaking, for balanced parameters, we achieve speed up ~2x speed up for verification. For short signature parameters, we achieve ~4x speed up for verification. The signing time is slightly (between 6% to 25 %) faster. The key gen time is slightly (between 8% to 10%) slower for some parameter sets, and faster for other parameter sets.
>   While the addition of the invertibility check was supposed to slow down the verification, we achieved a speed-up at last due to a revamped implementation (but still generic) for AVX2.
>
> We thank Markku-Juhani for his nice attack, and Ward for his timely suggestion on understanding this attack.
>
> We welcome more scrutiny and examinations of ALTEQ.
>
> Best regards,
> Youming (on behalf of the ALTEQ team)

On Monday, 4 September 2023 at 10:12:00 am UTC+10 Youming Qiao wrote:

> Dear Markku-Juhani,
>
> Thank you again for your close examination of ALTEQ. We took your attack seriously, and we had an updated version about three weeks ago.
>
> We delayed the release, because we wished to examine other possible attacks following your approach more carefully, and took this opportunity to speedup the code further.
>
> We aim to release a new version, hopefully sometime this week, and if not, definitely next week. Thank you for your patience.
>
> Best regards,
> Youming

From: Lars Ran <lars95ran@gmail.com>
Sent: Thursday, March 7, 2024 4:07 AM
To: pqc-comments
Cc: pqc-forum
Subject: Round 1 (Additional Signatures) OFFICIAL COMMENT: ALTEQ

Dear all,

We would like to draw your attention to our analysis on the ALTEQ signature scheme available at https://eprint.iacr.org/2024/364/ and published at CBCrypto23.
We devise an algebraic algorithm for solving the ATFE problem that results from modelling ATFE as a matrix code equivalence (MCE) problem.
Our algorithm has implications on the security of ALTEQ and is summarized as follows. (Table taken from the paper).

Table 1. Comparison of the concrete complexities (in $\log_2$ s
for solving ATFE. The superscript ^{a} means that the estir
algebraic model, and ^{b} from a graph-based birthday mod
includes the cost of field operations estimated as $O(\lceil \log_2 q$

NIST Sec. level	n	q	Tang et al. [TDJ+22]	Beullens [Beu22]	ALT [B
—	9	$2^{18}-1$	133	38	
—	10	$2^{17}-1$	133	122	
—	11	$2^{16}-5$	138	85	

We have timely communicated our results to the ALTEQ team.
As a result of our work, the ALTEQ signature scheme requires re-parametrization for all three NIST security levels.

Best regards,
Lars Ran, Simona Samardjiska, and Monika Trimoska

From: Youming Qiao <jimmyqiao86@gmail.com>
Sent: Thursday, March 7, 2024 5:56 AM
To: pqc-forum <pqc-forum@list.nist.gov>
Cc: Lars Ran <lars95ran@gmail.com>; pqc-forum <pqc-forum@list.nist.gov>; pqc-comments <pqc-comments@nist.gov>
Subject: Re: Round 1 (Additional Signatures) OFFICIAL COMMENT: ALTEQ

Dear Lars, Simona, and Monika, and colleagues on the forum,

First, thanks to Lars, Simona and Monica for discovering this new algorithm, and their generous help with understanding it.

However, we wish to point out that the analysis of this algorithm is based on an assumption, which is not supported by experiment data. That is Assumption 1 in the paper, which assumes that a certain family of syzygies are all the syzygies. It is the key to the analysis of the solving degree.

In Table 4 of the paper (https://eprint.iacr.org/2024/364/), experiments computing the syzygy dimensions give the following.

Table 4. Experimental syzygies.

n	$d=3$		$d=4$	
	experiment	prediction	experiment	prediction
5	16	16	906	700
6	72	35	4149	2691
7	64	64	7889	7889
8	105	105		
9	160	160		
10	231	231		
11	320	320		

However, we computed the syzygy dimensions for some larger n and d, and the results are as follows. (These have been communicated to the MEDS team.)

n	d=3, experiment	d=3, prediction	d=4, experiment	d=4, prediction	d=5, experiment	d=5, prediction
5	16	16	906	700		
6	72	35	4149	2691		
7	64	64	7889	7889	**314546**	**305025**
8	105	105	**20386**	**19440**		

The red font indicates where Assumption 1 is violated. The red bold font indicates the new experiment data not present in the previous table.

So we see that Assumption 1 fails for all n from 5 to 8, namely there are more syzygies than predicted by Assumption 1. Since this assumption is the key to establish the complexity estimate in the paper, its failure for these n suggests that more theoretical or experimental support is needed to claim the solving

degrees as in the paper.

To obtain further data for larger n/d of interest, such as n/d=9/5 or 10/4, requires computation power not accessible to us. Also note that if the solving degree for n=13 is 10 instead of 9 (as reported in the paper), then the bit complexity of the algorithm would be 128 for n=13.

As a result, we believe that the actual behaviour of this new modelling is an interesting open problem. To start with, it would be nice to study the new syzygies that appear in the experiments for n/d=7/5 and 8/4.

We welcome more comments, suggestions, and efforts from the community in understanding this algorithm, especially the differences between experimental and predicted syzygies.

Best regards,
Youming Qiao (on behalf of the ALTEQ team)

From: simona s <simona.samardziska@gmail.com>
Sent: Friday, March 8, 2024 6:04 AM
To: Youming Qiao
Cc: pqc-forum; pqc-comments
Subject: Re: [pqc-forum] Re: Round 1 (Additional Signatures) OFFICIAL COMMENT: ALTEQ

Dear Youming, dear all,

We first thank Youming for sharing his point of view on the attack. We appreciate it.

However, we would like to point out a few inaccuracies in the statement and give more explanation.
1. The attack was done only by the three of us, and not the MEDS team, although we are all part of the MEDS team.
2. Table 4 from our paper as presented by Youming is incomplete, two entries are missing, and they have been added to the table (and communicated to Youming) The table looks as follows

Table 4. Experimental syzygies.

n	$d=3$		$d=4$	
	experiment	prediction	experiment	prediction
5	16	16	906	700
6	72	35	4149	2691
7	64	64	7889	7889
8	105	105	20386	19440
9	160	160	42363	42363
10	231	231		
11	320	320		

It can be seen that we have experimentally confirmed our prediction of the syzygies for n=9 and d=4. We believe the same trend continues for larger odd n. For even n, indeed more syzygies appear that our model does not account for. Still, in our opinion, it is a very small fraction that does not change the solving degree of the system.
3. We further believe that a significant portion of the unaccounted syzygies for small n come from the additional high structure of ATFEs in small dimensions, and their inherently larger automorphism group. This structure has been observed and confirmed in previous works including Beullens' attack on the "pre-version" of ALTEQ (https://eprint.iacr.org/2022/1528).
We expect that the discrepancy reduces for n>9. Unfortunately, n>9 are already very big parameters, and we were not able to experimentally test them on our available servers.
4. Even if the solving degree increases, compared to our estimate, it is unlikely to increase by much, and this is, in our opinion, only for the even n case. Recall that ALTEQ has n=13 for NIST level I. From our table in the paper (in terms of field operations) we see that n=13 would not be secure even if the degree of solving would be d=10 (namely n=17 and d=10 barely satisfies NIST level I, with cost of field operations included)

Table 2. Solving degrees and complexities for ATFE instances using the improved matrix-code modelling. The cost is given in terms of field operations.

n	d_{wit}	$\log_2$ cost attack
8	9	83
9	9	90
10	9	95
11	9	101
12	9	105
13	9	110
14	10	123
15	10	127
17	10	135
18	11	148
20	11	155
25	13	193
27	13	199
28	13	202
30	14	219
35	15	245
37	16	263
38	16	265
40	17	283

5. Finally, it is our recommendation that the ALTEQ team increase the parameters. Of course, they may choose not to follow this recommendation if they believe our analysis is incorrect, or our assumption too strong. We encourage the community to verify or disprove our analysis.

All the best,
Simona, Lars and Monika

On Thu, Mar 7, 2024 at 11:56 AM Youming Qiao <jimmyqiao86@gmail.com> wrote:

Dear Lars, Simona, and Monika, and colleagues on the forum,

First, thanks to Lars, Simona and Monica for discovering this new algorithm, and their generous help with understanding it.

However, we wish to point out that the analysis of this algorithm is based on an assumption, which is not supported by experiment data. That is Assumption 1 in the paper, which assumes that a certain family of syzygies are all the syzygies. It is the key to the analysis of the solving degree.

In Table 4 of the paper (https://eprint.iacr.org/2024/364/), experiments computing the syzygy dimensions give the following.

Table 4. Experimental syzygies.

n	$d=3$		$d=4$	
	experiment	prediction	experiment	prediction
5	16	16	906	700
6	72	35	4149	2691
7	64	64	7889	7889
8	105	105		
9	160	160		
10	231	231		
11	320	320		