From: Yongge Wang <yongge.wang@gmail.com>
Sent: Sunday, December 24, 2017 12:59 PM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: HK17

Dear HK17 Designer and all,

I have two comments on this submission.

1. This submission is a Key Agreement Protocol (DH style) and seems not fall into any of the NIST PQC CFP categories (pk signature, pk encryption, kem).

2. I think the protocol may not be correct (or am I missing something?) The protocol arithmetic operations are over quaternion/octonion. So we need to be aware of the fact that the quaternions are not commutative and the octonions are neither commutative nor associative. In the protocol specification, Section 3.1 (Computing Session Keys step i), it claims that k_A=k_B. The proof for this correctness fact is as follows:

k_A = f'(q_A)^m . h'(q_A)^r . q_B . h'(q_A)^s . f'(q_A)^n = h'(q_A)^r . f'(q_A)^m . q_B . f'(q_A)^n . h'(q_A)^s = k_B

In the above proof, obviously the "commutative" property is used. As we have mentioned, the commutative property does not hold for quaternion (neither for octonion). So the correctness proof is invalid.

The designer has given some numerical examples.. I did not check it.. but I am not sure why the numerical example actually is correct without the commutative property.

thanks!
Yongge

From: Yongge Wang <yongge.wang@gmail.com>
Sent: Sunday, December 24, 2017 1:25 PM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: HK17

Furthermore, the implementation may not be correct since the proposal claims
that in the implementation, "no big number libraries needed."
(both implementation contains no big number libraries)

It should be noted that in the computation of
f'(q_A)^m . h'(q_A)^r . q_B . h'(q_A)^s . f'(q_A)^n we have f'(q_A)=(a,b,c,d) where
a/b/c/d are in the format of 0.xxxx and the polynomial f has degree d=16 or 32.
Since m is any integer (e.g., 245 as in the numerical example). Thus f'(q_A)^m
is in the format of XXX.xxxxxx...xxxx, where there may be 245x16x4=15680
fractional digits... this will obviously not work on any 32-bit CPUs if one does not use any special numeric package. In the proposal, the examples round each number to 4-fractional digits during computation... but this will make the equation non-valid (the round takes places at different steps at Alice or Bob side).

thanks!
Yongge

On Sun, Dec 24, 2017 at 8:59 PM, Yongge Wang <yongge.wang@gmail.com> wrote:

> Dear HK17 Designer and all,
>
> I have two comments on this submission.
>
> 1. This submission is a Key Agreement Protocol (DH style) and seems not fall into any of the NIST PQC CFP categories (pk signature, pk encryption, kem).
>
> 2. I think the protocol may not be correct (or am I missing something?) The protocol arithmetic operations are over quaternion/octonion. So we need to be aware of the fact that the quaternions are not commutative and the octonions are neither commutative nor associative. In the protocol specification, Section 3.1 (Computing Session Keys step i), it claims that k_A=k_B. The proof for this correctness fact is as follows:
>
> k_A = f'(q_A)^m . h'(q_A)^r . q_B . h'(q_A)^s . f'(q_A)^n = h'(q_A)^r . f'(q_A)^m . q_B . f'(q_A)^n . h'(q_A)^s = k_B
>
> In the above proof, obviously the "commutative" property is used. As we have mentioned, the commutative property does not hold for quaternion (neither for octonion). So the correctness proof is invalid.
>
> The designer has given some numerical examples.. I did not check it.. but I am not sure why the numerical example actually is correct without the commutative property.
>
> thanks!
> Yongge

Kerman, Sara J. (Fed)

From: D. J. Bernstein <djb@cr.yp.to>
Sent: Monday, December 25, 2017 7:01 AM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] OFFICIAL COMMENT: HK17

I don't see any secret randomness in crypto_kem_enc() in the reference
HK17 implementation. It's easy to check this: run crypto_kem_enc() twice and see if the results are identical. If there is in fact no secret randomness then a break of the code is trivial.

My impression is that the intended KEM (obtained in the usual way from the specified DH) requires the following change to the data flow in the reference implementation: the "m", "n", and "polynomial" items obtained from "pk" in crypto_kem_enc(), which in turn are generated randomly by crypto_kem_keypair(), should instead be generated randomly in crypto_kem_enc().

Of course it's the responsibility of the submitters to clearly specify the KEM and to make sure that the reference implementation matches.

Yongge Wang writes:
> In the above proof, obviously the "commutative" property is used. As
> we have mentioned, the commutative property does not hold for
> quaternion (neither for octonion). So the correctness proof is invalid.

Tanja Lange and I have been discussing this, and our impression is that the stated identity follows from the Moufang identities. It's important to note that f and h are evaluated at the same quaternion or octonion.
The submitters should spell out the details.

> f’(q_A)^m is in the format of XXX.xxxxxx...xxxx, where there may be
> 245x16x4=15680 fractional digits...

The implementation reduces modulo (e.g.) 251 at each step, and this is compatible with the definition of the shared secret.

The above comments should not be viewed in any way as any sort of endorsement of the security of HK17.

---Dan

From: Yongge Wang <yongge.wang@gmail.com>
Sent: Monday, December 25, 2017 9:59 AM
To: pqc-forum@list.nist.gov; pqc-comments
Subject: Re: [pqc-forum] OFFICIAL COMMENT: HK17

On Mon, Dec 25, 2017 at 3:00 PM, D. J. Bernstein <djb@cr.yp.to> wrote:

> Tanja Lange and I have been discussing this, and our impression is that
> the stated identity follows from the Moufang identities. It's important
> to note that f and h are evaluated at the same quaternion or octonion.
> The submitters should spell out the details.

thanks for pointing out this.. that is right.. if both f and h evaluate on the same quaternion or octonion, then the commutative law works there..

> > f'(q_A)^m is in the format of XXX.xxxxxx...xxxx, where
> > there may be 245x16x4=15680 fractional digits...
> The implementation reduces modulo (e.g.) 251 at each step, and this is
> compatible with the definition of the shared secret.

OK, this may be one potential interpretation of the missing part in the proposal. But it may not be something that the submitter has in mind? From the numerical example, it shows that each time, when secret is derived using mod operation, all the fractional part is simply dropped (not rounded)... But the computation of r_A/r_B in the numerical example has four digit fractional part. So I think during the intermediate steps for computing r_A/r_B, even if mod operation is done, it is different from the mod operation in the secret derivation step.

E.g., in the K_B computation process, 0.3184 mod 215 = 79. This is obtained by using the fact 0.3184x251=79.9184. Note that here 0.9184 is dropped completely..

Even if the mod operation is defined correctly
in the intermediate steps, there are issues... e.g.,
(2*0.1021) mod 251 != 2*(0.1021 mod 251) mod 251

the reason is
(2*0.1021) mod 251 =0.2042*251 mod 251 = 51.2291 mod 251 = 51
2*(0.1021 mod 251) mod 251 = 2*(0.1021*251 mod 251) mod 251=50

thank!
Yongge

From: D. J. Bernstein <djb@cr.yp.to>
Sent: Monday, December 25, 2017 12:16 PM
To: pqc-forum@list.nist.gov
Cc: pqc-comments
Subject: OFFICIAL COMMENT: HK17

Dear designers, dear all,

The following attack script breaks HK17 for all proposed parameters.
Specifically, this script breaks the Python key-exchange implementation included in the HK17 submission. This key-exchange implementation appears to match the intent of the HK17 documentation, except that the documentation includes a normalization step; this step does not affect the attack.

This attack takes p+1 simple computations (and is a search so Grover's algorithm is applicable, but all proposed parameters are small enough to be broken by a non-quantum attack). For comparison, the submission says

* 2^64 pre-quantum security for p=251,
* 2^128 pre-quantum security for p=65521,
* 2^256 pre-quantum security for p=4294967291, and
* 2^512 pre-quantum security for p=18446744073709551557.

Our attack takes about 2^8, 2^16, 2^32, and 2^64 simple computations for these parameter sets.

For simplicity the attack script focuses on the case that the public key rA is invertible, which occurs almost all of the time. Slightly more work should be able to handle the occasional exceptions.

To use this script, save the following Python code as break.py; copy octonions.py from the HK17 submission to octonions.py in this directory; copy HK17-O.py from the HK17 submission to ref.py in this directory; and run "python break.py".

---Daniel J. Bernstein and Tanja Lange

```
import octonions
import ref
import sys

p = ref.modulo
print ref.message
print ref.times

print "eve observes public parameters and alice's public key:"
print 'oa =',ref.oa
print 'ob =',ref.ob
print 'rA =',ref.rA

def modprecip(x):
  x %= p
```

```
  if x == 0: raise Exception('dividing by 0')
  return pow(x,p-2,p)

def octonionrecip(x):
  xnormsq = sum(xi**2 for xi in x) % p
  xconj = (x[0],-x[1],-x[2],-x[3],-x[4],-x[5],-x[6],-x[7])
  return octonions.scale(xconj,modprecip(xnormsq),p)

try:
  rArecip = octonionrecip(ref.rA)
except:
  raise Exception('public key is not invertible, skipping this case for simplicity')

try:
  obrecip = octonionrecip(ref.ob)
except:
  raise Exception('ob is not invertible, should have caught from rA test') assert octonions.multiply(obrecip,ref.ob,p) ==
(1,0,0,0,0,0,0,0)

# goal: write rA as x*ob*y for some x,y in \F_p[oa] # this forces x to be in \F_p + oa\F_p # wlog take x to be 1 or in \F_p +
oa for i in range(p):
  for j in [0,1]:
    if j == 0 and i != 1: continue
    x0 = (i,0,0,0,0,0,0,0)
    x1 = octonions.scale(ref.oa,j,p)
    x = octonions.summ(x0,x1,p)
    try:
      xrecip = octonionrecip(x)
    except:
      continue
    t = octonions.multiply(xrecip,ref.rA,p)
    # goal: write t as ob*y for some y in \Z[oa]
    y = octonions.multiply(obrecip,t,p)
    if octonions.multiply(y,ref.oa,p) == octonions.multiply(ref.oa,y,p):
      print "eve's secret key:",x,y
      print "now eve looks at bob's ciphertext (DH public key):"
      print 'rB =',ref.rB
      k = octonions.multiply(x,octonions.multiply(ref.rB,y,p),p)
      print "eve's session key =",k
      print 'now peek at secrets to verify attack worked:'
      print "alice's session key =",ref.k1
      print "bob's session key =",ref.k2
      sys.exit(0)
```

From: Yongge Wang <yongge.wang@gmail.com>
Sent: Tuesday, December 26, 2017 3:04 AM
To: pqc-comments; pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] OFFICIAL COMMENT: HK17

After reading Bernstein-Lange attack, one may wonder whether the scheme could be made secure by choosing a large enough prime p (e.g., 1000 bit p). Unfortunately, no matter how big the p it is, one can break the protocol by solving a homogeneous quadratic equation system of eight equations in four unknowns. I believe this could be done in constant steps(?) using Kipnis and Shamir's relinearization techniques or the Gröbner basis algorithm or F4/F5 algorithms). For details, see https://webpages.uncc.edu/yonwang/octonionDH.pdf

In case I am wrong, please let me know. thanks!
Yongge

On Mon, Dec 25, 2017 at 8:15 PM, D. J. Bernstein <djb@cr.yp.to> wrote:

> Dear designers, dear all,
>
> The following attack script breaks HK17 for all proposed parameters.
> Specifically, this script breaks the Python key-exchange implementation
> included in the HK17 submission. This key-exchange implementation
> appears to match the intent of the HK17 documentation, except that the
> documentation includes a normalization step; this step does not affect
> the attack.
>
> This attack takes p+1 simple computations (and is a search so Grover's
> algorithm is applicable, but all proposed parameters are small enough to
> be broken by a non-quantum attack). For comparison, the submission says
>
> * 2^64 pre-quantum security for p=251,
> * 2^128 pre-quantum security for p=65521,
> * 2^256 pre-quantum security for p=4294967291, and
> * 2^512 pre-quantum security for p=18446744073709551557.
>
> Our attack takes about 2^8, 2^16, 2^32, and 2^64 simple computations for
> these parameter sets.
>
> For simplicity the attack script focuses on the case that the public key
> rA is invertible, which occurs almost all of the time. Slightly more
> work should be able to handle the occasional exceptions.
>
> To use this script, save the following Python code as break.py; copy
> octonions.py from the HK17 submission to octonions.py in this directory;
> copy HK17-O.py from the HK17 submission to ref.py in this directory; and
> run "python break.py".
>
> ---Daniel J. Bernstein and Tanja Lange

From: perret <ludovic.perret@lip6.fr>
Sent: Tuesday, December 26, 2017 6:23 AM
To: Yongge Wang
Cc: pqc-comments; pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] OFFICIAL COMMENT: HK17

Dear Yongge,

> Le 26 déc. 2017 à 09:03, Yongge Wang <yongge.wang@gmail.com> a écrit :
>
> After reading Bernstein-Lange attack, one may wonder whether the
> scheme could be made secure by choosing a large enough prime p (e.g.,
> 1000 bit p). Unfortunately, no matter how big the p it is, one can
> break the protocol by solving a homogeneous quadratic equation system
> of eight equations in four unknowns. I believe this could be done in
> constant steps(?) using Kipnis and Shamir's relinearization techniques
> or the Gröbner basis algorithm or F4/F5 algorithms). For details, see
>
> https://webpages.uncc.edu/yonwang/octonionDH.pdf
>
In case I am wrong, please let me know. thanks!

>
>

In general, such a small system can be indeed solved very easily.
Do you have a code for generating these equations ?

Best Regards,

Ludovic Perret
Université Pierre et Marie Curie
Tel : 01 44 27 87 59
web : http://www-polsys.lip6.fr/~perret/

> Yongge
>
> On Mon, Dec 25, 2017 at 8:15 PM, D. J. Bernstein <djb@cr.yp.to> wrote:
> Dear designers, dear all,
>
> The following attack script breaks HK17 for all proposed parameters.
> Specifically, this script breaks the Python key-exchange implementation
> included in the HK17 submission. This key-exchange implementation
> appears to match the intent of the HK17 documentation, except that the

Kerman, Sara J. (Fed)

From: D. J. Bernstein <djb@cr.yp.to>
Sent: Tuesday, December 26, 2017 7:38 AM
To: pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] OFFICIAL COMMENT: HK17

perret writes:
> In general, such a small system can be indeed solved very easily.
> Do you have a code for generating these equations ?

It's simply the equation rA = x*ob*y mentioned in our attack script, where x = x0 + x1 oa and y = y0 + y1 oa. Here oa,ob,rA are public octonions over F_p, and x0,x1,y0,y1 are the variables in F_p.

There are really only three variables, since the equation is projective; as the attack script says, "wlog take x to be 1 or in \F_p + oa".

---Dan

From: Yongge Wang <yongge.wang@gmail.com>
Sent: Tuesday, December 26, 2017 1:49 PM
To: perret
Cc: pqc-comments; pqc-forum
Subject: Re: [pqc-forum] OFFICIAL COMMENT: HK17

Dear Ludovic,
thanks for the message.

> In general, such a small system can be indeed solved very easily.

thanks for this confirmation.

> Do you have a code for generating these equations ?

I do not have a python or C script.. (I am generally lazy in writing code). But I just revised the document at https://webpages.uncc.edu/yonwang/octonionDH.pdf . It now has the exact formula to build the homogeneous quadratic equation system using the publicly observed values r_A, r_B, o_A, o_B. In the current draft, these are equation (11) and equation (12).. So if one is good at python, one can quickly convert them to python (I have limited python knowledge). Then one may use the Gröbner basis algorithm or F4/F5 algorithms to solve the equation (again, I do not have experience with these packages.. so did not try).

Thanks!
Yongge

On Tue, Dec 26, 2017 at 2:23 PM, perret <ludovic.perret@lip6.fr> wrote:

> Dear Yongge,
>
> > Le 26 déc. 2017 à 09:03, Yongge Wang <yongge.wang@gmail.com> a écrit :
> >
> > After reading Bernstein-Lange attack, one may wonder whether the scheme could be made
> > secure by choosing a large enough prime p (e.g., 1000 bit p). Unfortunately, no matter how big the p it is,
> > one can break the protocol by solving a homogeneous quadratic equation system of eight equations
> > in four unknowns. I believe this could be done in constant steps(?) using Kipnis and Shamir's relinearization
> > techniques or the Gröbner basis algorithm or F4/F5 algorithms). For details, see
> > https://webpages.uncc.edu/yonwang/octonionDH.pdf
> >
> > In case I am wrong, please let me know. thanks!
>
> In general, such a small system can be indeed solved very easily.
> Do you have a code for generating these equations ?
>
> Best Regards,
>
> Ludovic Perret
> Université Pierre et Marie Curie
> Tel : 01 44 27 87 59
> web : http://www-polsys.lip6.fr/~perret/

Kerman, Sara J. (Fed)

From: Yanbin Pan <panyanbin@amss.ac.cn>
Sent: Wednesday, December 27, 2017 5:31 AM
To: pqc-comments
Cc: pqc-forum
Subject: OFFICIAL COMMENT: HK17
Attachments: HK17.pdf

Dear Dr. Chen, Moody, and Liu,

We group find that the key exchange scheme HK17 is not secure. Any passive adversary can recover the shared key very efficiently.

The key observation is that any octonion (or quaternion) satisfies a quadratic equation, so for any octonion o_A, and any polynomial g(x), we can find a,b such that g(o_A)= a o_A+b.

For HK17, since r_A=f(o_A)^m o_B f(o_A)^n, we can write r_A=(a o_A+b)o_B (c o_A+d) where a,b,c,d are unknowns. By solving a system of linear equations over Z_p, we can find a solution a, b, c d.
Next we can prove that (a o_A+b)r_B (c o_A+d) equals the shared key.

See the attachment for more details.

Best regards,
Yanbin

Cryptanalysis of HK17

Haoyu Li[1,2], Renzhang Liu[3], Yanbin Pan[1], Tianyuan Xie[1,2]

[1]Key Laboratory of Mathematics Mechanization, NCMIS,
Academy of Mathematics and Systems Science, Chinese Academy of Sciences
Beijing 100190, China
[2] School of Mathematical Sciences, University of Chinese Academy of Sciences,
Beijing 100049, China
[3] State Key Laboratory of Information Security, Institute of Information Engineering,
Chinese Academy of Sciences, Beijing, China.

Abstract. Very recently, a key exchange scheme called HK17 was submitted to NIST as a candidate of the standard of post-quantum cryptography. The HK17 scheme employs some hypercomplex numbers as the basic objects, such as quaternions and octonions. In this paper, we show that HK17 is insecure since a passive adversary can recover the shared key in polynomial time.

1 Introduction

In December of 2017, NIST published the Round 1 submissions for the Post-Quantum Cryptography. Among all the candidate schemes, a key exchange called HK17 was proposed by Hecht and Kamlofsky [1]. Different from most of the popular schemes, the HK17 scheme uses hypercomplex numbers, such as quanternions and octonions.

Some strong points of HK17 were also pointed out in [1], such as: using ordinary modular arithmetic but without big number libraries, relatively fast operation, non-associativity of products and powers, parametric security levels, no classical nor quantum attack at sight, possible resistance to side-channel attacks, easy firmware migration and conjectured semantical security IND-CCA2 compliance.

However, in this paper, we will show that the HK17 scheme is not secure. More precisely, any passive adversary can recover the shared key very efficiently.

2 Preliminaries

2.1 Quaternions and Octonions

In mathematics, Quaternions and Octonions are generalization of the complex numbers. Quaternions are the noncommutative generalization of the complex numbers. In general, a quaternion can be represented in the following form:

$$a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}$$

where a, b, c, d are all real numbers and $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ are the fundamental quaternion units. Furthermore, the units $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ satisfy the following identities:

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1.$$

The octonions are nonassociative generalization of quaternions. Generally speaking, an octonions $\boldsymbol{o}$ can be represented as a real linear combination of the unit octonions:

$$\boldsymbol{o} = a_0\boldsymbol{e}_0 + a_1\boldsymbol{e}_1 + \cdots + a_7\boldsymbol{e}_7$$

where $\boldsymbol{e}_0$ is the real number 1. Furthermore, the product of each pair of terms can be given by multiplication of the coefficients and a multiplication table of the unit octonions, like the following:

$$\boldsymbol{e}_i\boldsymbol{e}_j = \begin{cases} \boldsymbol{e}_i & i = 0 \\ \boldsymbol{e}_j & j = 0 \\ -\delta_{ij}\boldsymbol{e}_0 + \epsilon_{ijk}\boldsymbol{e}_k & \text{otherwise} \end{cases}$$

where δ_{ij} is the Kronecker delta and $\epsilon_{ijk} = 1$ when $ijk = 123$, 145, 176, 246, 257, 357, 347, 365.

2.2 HK17

The HK17 Key Exchange scheme uses some hypercomplex numbers such as quaternions and octonions. We take the octonions version as an example to describe this scheme.

* Initialization:
  1) Alice choose two non-zero octonions $\boldsymbol{o}_A$, $\boldsymbol{o}_B$ with each coordinate uniformly in $\mathbb{Z}_p$ with some prime p;
  2) Alice choose two integers m, n and a non-zero polynomial $f(x) \in \mathbb{Z}_p[x]$ with degree d such that $f(\boldsymbol{o}_A) \neq 0$, and (f, m, n) is Alice's private key;
  3) Alice send $\boldsymbol{o}_A$ and $\boldsymbol{o}_B$ to Bob;
  4) Bob choose two integers r, s and a non-zero polynomial $h(x) \in \mathbb{Z}_p[x]$ with degree d such that $h(\boldsymbol{o}_A) \neq 0$, and (h, r, s) is Alice's private key.
* Computing the tokens:
  1) Alice compute the value $\boldsymbol{r}_A = f(\boldsymbol{o}_A)^m\boldsymbol{o}_B f(\boldsymbol{o}_A)^n$ and send it to Bob;
  2) Bob compute the value $\boldsymbol{r}_B = h(\boldsymbol{o}_A)^r\boldsymbol{o}_B h(\boldsymbol{o}_A)^s$ and send it to Alice.
* Computing Session Keys:
  1) Alice compute her key: $K_A = f(\boldsymbol{o}_A)^m\boldsymbol{r}_B f(\boldsymbol{o}_A)^n$;
  2) Bob compute his key: $K_B = h(\boldsymbol{o}_A)^r\boldsymbol{r}_A h(\boldsymbol{o}_A)^s$.

It can be verified that

$$\begin{aligned} K_A &= f(\boldsymbol{o}_A)^m\boldsymbol{r}_B f(\boldsymbol{o}_A)^n \\ &= f(\boldsymbol{o}_A)^m h(\boldsymbol{o}_A)^r\boldsymbol{o}_B h(\boldsymbol{o}_A)^s f(\boldsymbol{o}_A)^n \\ &= h(\boldsymbol{o}_A)^r f(\boldsymbol{o}_A)^m\boldsymbol{o}_B f(\boldsymbol{o}_A)^n h(\boldsymbol{o}_A)^s \\ &= h(\boldsymbol{o}_A)^r\boldsymbol{r}_A h(\boldsymbol{o}_A)^s \\ &= K_B \end{aligned}$$

Finally, Alice and Bob share the common key $K_A = K_B$.

3 Our Attack against the Octonions Version of HK17

3.1 The key observation

We have the following key observations.

Lemma 1. *For any octonion $\boldsymbol{o}$, we can find α, β in polynomial time such that*

$$\boldsymbol{o}^2 + \alpha\boldsymbol{o} + \beta = 0.$$

Furthermore, when all the coordinates of $\boldsymbol{o}$ are in $\mathbb{Z}_p$, for any polynomial $g(x) \in \mathbb{Z}_p[x]$, there exist a, $b \in \mathbb{Z}_p$, such that

$$g(\boldsymbol{o}) = a\boldsymbol{o} + b.$$

Proof. Given an octonion $\boldsymbol{o} = a_0\boldsymbol{e}_0 + a_1\boldsymbol{e}_1 + \cdots + a_7\boldsymbol{e}_7$, we have

$$\begin{aligned}&(a_0\boldsymbol{e}_0 + a_1\boldsymbol{e}_1 + \cdots + a_7\boldsymbol{e}_7)^2\\ =&(a_0^2 - a_1^2 - \cdots - a_7^2)\boldsymbol{e}_0 + 2a_0a_1\boldsymbol{e}_1 + \cdots + 2a_0a_7\boldsymbol{e}_7\\ =&2a_0(a_0\boldsymbol{e}_0 + \cdots + a_7\boldsymbol{e}_7) - (a_0^2 + \cdots + a_7^2)\boldsymbol{e}_0\end{aligned}$$

Let

$$\alpha = -2a_0, \beta = a_0^2 + \cdots + a_7^2.$$

Then $\boldsymbol{o} = a_0\boldsymbol{e}_0 + a_1\boldsymbol{e}_1 + \cdots + a_7\boldsymbol{e}_7$ is a solution of

$$x^2 + \alpha x + \beta = 0.$$

Then for any polynomial $g(x) \in \mathbb{Z}_p[x]$, we can write

$$g(x) = (x^2 + \alpha x + \beta)q(x) + (ax + b),$$

which implies immediately that

$$g(\boldsymbol{o}) = a\boldsymbol{o} + b.$$

Lemma 2. *For HK17, given $\boldsymbol{o}_A$, $\boldsymbol{o}_B$, $\boldsymbol{r}_A$, there exists a polynomial time (in $\log p$) algorithm to find $a, b, c, d \in \mathbb{Z}_p$ such that*

$$\boldsymbol{r}_A = (a\boldsymbol{o}_A + b)\boldsymbol{o}_B(c\boldsymbol{o}_A + d).$$

Proof. By Lemma 1, we know that there exist $a, b, c, d \in \mathbb{Z}_p$, such that

$$f(\boldsymbol{o}_A)^m = a\boldsymbol{o}_A + b, \text{ and } f(\boldsymbol{o}_A)^n = c\boldsymbol{o}_A + d.$$

Therefore, we can write

$$\begin{aligned}\boldsymbol{r}_A &= f(\boldsymbol{o}_A)^m\boldsymbol{o}_B f(\boldsymbol{o}_A)^n\\ &= (a\boldsymbol{o}_A + b)\boldsymbol{o}_B(c\boldsymbol{o}_A + d)\\ &= ac\boldsymbol{o}_A\boldsymbol{o}_B\boldsymbol{o}_A + ad\boldsymbol{o}_A\boldsymbol{o}_B + bc\boldsymbol{o}_B\boldsymbol{o}_A + bd\boldsymbol{o}_B\end{aligned}$$

By comparing every corresponding coordinate of $\boldsymbol{r}_A$ and $ac\boldsymbol{o}_A\boldsymbol{o}_B\boldsymbol{o}_A + ad\boldsymbol{o}_A\boldsymbol{o}_B + bc\boldsymbol{o}_B\boldsymbol{o}_A + bd\boldsymbol{o}_B$, we will have eight linear equations with four unknowns ac, ad, bc, bd. By the existence of a, b c, d, we can always solve the system of the eight linear equations to get a solution (s_1, s_2, s_3, s_4) for (ac, ad, bc, bd).

Note that since a, b can not be zero at the same time if $\boldsymbol{r}_A \neq 0$, so we can tell from which is nonzero. For example if $s_1 = 0$ and $s_2 = 0$, then b must not be zero. Similarly, we can also know that if c or d is zero or not. Without loss of generality, assume $a \neq 0$, $c \neq 0$, then we can set $a = 1$, and

$$(1, s_1^{-1}s_3, s_1, s_2)$$

must be a solution, since

$$\begin{aligned}\boldsymbol{r}_A &= (a\boldsymbol{o}_A + b)\boldsymbol{o}_B(c\boldsymbol{o}_A + d)\\ &= a(\boldsymbol{o}_A + a^{-1}b)\boldsymbol{o}_B(c\boldsymbol{o}_A + d)\\ &= (\boldsymbol{o}_A + a^{-1}b)\boldsymbol{o}_B(ac\boldsymbol{o}_A + ad).\end{aligned}$$

Note that we can also set a to be any nonzero element in $\mathbb{Z}_p$ and solve the other corresponding b, c, d.

Lemma 3. *For HK17 key exchange scheme, if we can find any two polynomial $g_1(x), g_2(x) \in \mathbb{Z}_p[x]$, such that*

$$\boldsymbol{r}_A = g_1(\boldsymbol{o}_A)\boldsymbol{o}_B g_2(\boldsymbol{o}_A),$$

then the shared key

$$K = g_1(\boldsymbol{o}_A)\boldsymbol{r}_B g_2(\boldsymbol{o}_A).$$

Proof. Note that

$$\begin{aligned}K_B &= h(\boldsymbol{o}_A)^r \boldsymbol{r}_A h(\boldsymbol{o}_A)^s\\ &= h(\boldsymbol{o}_A)^r g_1(\boldsymbol{o}_A)\boldsymbol{r}_B g_2(\boldsymbol{o}_A) h(\boldsymbol{o}_A)^s\\ &= g_1(\boldsymbol{o}_A) h(\boldsymbol{o}_A)^r \boldsymbol{r}_B h(\boldsymbol{o}_A)^s g_2(\boldsymbol{o}_A)\\ &= g_1(\boldsymbol{o}_A)\boldsymbol{r}_B g_2(\boldsymbol{o}_A)\\ &= K.\end{aligned}$$

The lemma follows.

3.2 Our Attack

Based on the lemmas above, we present our attack.

Step 1 When the adversary gets $\boldsymbol{o}_A$, $\boldsymbol{o}_B$, $\boldsymbol{r}_A$ by eavesdropping, he can compute $a, b, c, d \in \mathbb{Z}_p$ such that

$$\boldsymbol{r}_A = (a\boldsymbol{o}_A + b)\boldsymbol{o}_B(c\boldsymbol{o}_A + d),$$

by Lemma 2.

Step 2 Compute

$$K = (a\boldsymbol{o}_A + b)\boldsymbol{r}_B(c\boldsymbol{o}_A + d).$$

By Lemma 3, we know K is exactly the shared key established by Alice and Bob.

3.3 Experimental Result

We take the example on Page 11 in [1] to verify our attack. In the example, we have

- $p = 251$;
- $\boldsymbol{o}_A = (157, 188, 177, 188, 203, 149, 217, 148)$;
- $\boldsymbol{o}_B = (40, 207, 6, 33, 75, 79, 98, 54)$;
- $\boldsymbol{r}_A = (121, 3, 110, 243, 184, 230, 202, 171)$;
- $\boldsymbol{r}_B = (90, 42, 17, 119, 150, 23, 110, 182)$.

After Step 1 in our attack, we find the solution $(1, 142, 75, 187)$ such that

$$\boldsymbol{r}_A = (\boldsymbol{o}_A + 142)\boldsymbol{o}_B(75\boldsymbol{o}_A + 187).$$

After Step 2, we recover the shared key

$$K = (\boldsymbol{o}_A + 142)\boldsymbol{r}_B(75\boldsymbol{o}_A + 187) = (84, 242, 130, 31, 84, 244, 45, 20),$$

which is exactly the shared key established in [1].

4 Our Attack against the Quaternions Version of HK17

In [1], a quaternions version was also proposed, which has the same framework to the octonions version, but with an additional normalization. It can be easily concluded that our attack can be extended to the quaternions version of HK17, since for any quaternions $\boldsymbol{q} = a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}$, it also satisfied a quadratic equation

$$x^2 - 2ax + (a^2 + b^2 + c^2 + d^2) = 0.$$

References

1. Hecht, Kamlofsky: HK17: Post Quantum Key Exchange Protocol Based on Hypercomplex Numbers. Available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/HK17.zip

From: qubit101 (gmail) <qubit101@gmail.com>
Sent: Thursday, December 28, 2017 2:40 PM
To: pqc-comments
Cc: Jorge Kamlofsky; pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: HK17

Dear Yongge,
(Yongge Wang)

Apologise, but I must correct your comments.

1. You said "1. This submission is a Key Agreement Protocol (DH style) and seems not fall into any of the NIST PQC CFP categories (pk signature, pk encryption, kem)."
That is irrelevant. See NIST-PQC-Submission Requirements (Note pg4 "previous NIST publications have tended to describe KEMs using the term “key agreement” (also known as the key exchange), and have tended to describe public key encryption schemes using the term “key transport.”). Otherwise, as an old published fact [1], any kind of OWTF serve to construct any kind of ass[ymetric protocol (Key exchange, Key transport, ElGamalCipher, ElGamal Signature, ZKP and so on). I myself published a way to achieve that [2] using Generalized Symmetric Decomposition (GSD) as OFTW, exactly like in this PQC HK17 proposal. The fact is that you make one protocol and you easily derive the others.
[1] Baumslag G. in Designing Key Transport Protocols using Combinatorial Group Theory pp 35 in L. Gerritzen et al (Editors), Algebraic Methods in Cryptography, Contemporary Mathematics, AMS, Vol. 418, 2006
[2] “A Post-Quantum Set of Compact Asymmetric Protocols using a General Linear Group”, P. Hecht, Actas del VIII Congreso Iberoamericano de Seguridad Informática CIBSI’15, Ramió Aguirre J. el al (Eds), Universidad Politécnica de Madrid (España), 96-101 (2015) ISBN: 978-9978-301-61-6

2. You confuse non-commutativity of single arguments like quaternions or octonions with commutativity of their Polynomial powers. Two different private polynomial f(x), g(x) powers (m, n) does NOT commute if arguments (octonions o and o') are different but DO COMMUTE if arguments are the same, that means f(o)^m. g(o')^n != g(o')^n. f(o)^m but f(o)^m. g(o)^n = g(o)^n. f(o)^m. See i.e [3]. Therefore you should not be surprised that a common key is obtained at Alice and Bob sides. The arguments could be matrices and the protocol will work perfectly.
[3] Cao Z., Xiaolei D., Wang L.: New public-key cryptosystems using polynomials over non-commutative rings, Preprint arXiv/cr, eprint.iacr.org/2007/009.pdf (2007)

Thanks!
Peter Hecht

From: Peter Hecht <qubit101@gmail.com>
Sent: Thursday, December 28, 2017 4:43 PM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: HK17

Dear Yongge (and Dear All readers),

We studied your critics at https://webpages.uncc.edu/yonwang/octonionDH.pdf
and found a fatal and misleading error in it.

Your “attacked” protocol HK17-Octonions (Point 3. HK17) use simple polynomials of octonions and our proposal work clearly stated with secret powers of those polynomials.

We invite you to present a correct attack or rectify your conclusions.

Thanks!
Peter

Sent from Mail for Windows 10

From: Yongge Wang <yongge.wang@gmail.com>
Sent: Friday, December 29, 2017 3:33 PM
To: Peter Hecht
Cc: pqc-comments; pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] OFFICIAL COMMENT: HK17

Dear Peter,
thanks for the message. See my explanation:

> We studied your critics at https://webpages.uncc.edu/yonwang/octonionDH.pdf and
> found a fatal and misleading error in it. Your "attacked" protocol HK17-Octonions (Point 3. HK17)
> use simple polynomials of octonions and our proposal work (as clearly stated) with secret powers of those polynomials.

Original HK17:
In your scheme, the private key for Alice is two non-zero integers m,n and a polynomial f of degree d (d is secret)
the private key for Bob is two non-zero integers r,s and a polynomial g of degree d (d is secret but same as d for Alice)

To make the description simple, my reformulate your scheme as follows:

My version HK17:
the private key for Alice is two secret polynomials f1 and f2 (their degrees are secret)
the private key for Bob is two secret polynomials g1 and g2 (their degrees are secret)

Your original HK17 is a special case of my version since Alice can just take f1=f^m, f2=f^n
and Bob select g1=g^s, g2=g^t

that is, f1 is of degree md, f2 is of degree nd, g1 is of degree rd, and g2 is of degree sd.

Thanks.
Yongge

On Fri, Dec 29, 2017 at 6:16 PM, Peter Hecht <qubit101@gmail.com> wrote:

> Dear Yongge, Dear All,
>
> We studied your critics at https://webpages.uncc.edu/yonwang/octonionDH.pdf
>
> and found a fatal and misleading error in it.
>
> Your "attacked" protocol HK17-Octonions (Point 3. HK17) use simple polynomials of octonions and our proposal work (as clearly stated) with secret powers of those polynomials.

From: Peter Hecht <qubit101@gmail.com>
Sent: Friday, December 29, 2017 5:47 PM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: HK17

Dear Yongge, Dear All,

Thanks for your last comment.

The general idea of your critics seems to be sound, but our developer team does not give for granted how your attack would perform with a numerical example. It would not be the first time that theory differs from practical use. For that purpose, we want to see if 4. Break HK17 in O(1) steps [1] could recover with your alleged time complexity any KAT values provided by us, i.e. take our Alice/Bob values in any of this cases:

\PQC-HK17-Submission.zip\3 Optical Media\KAT\Examples with intermediate values\HK17-512bitsKeys.txt or

\PQC-HK17-Submission-updated.zip\3 Optical Media\KAT\Examples with intermediate values\HK17-256bitsKeys.txt

and obtain with your method Eve's recovered key. Your conclusions are impressive, let your arguments to be at the same level.

[1] https://webpages.uncc.edu/yonwang/octonionDH.pdf

Thanks

Peter

Sent from Mail for Windows 10

Kerman, Sara J. (Fed)

From: D. J. Bernstein <djb@cr.yp.to>
Sent: Monday, January 01, 2018 12:48 AM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: HK17

Here's a faster attack script against HK17.

The previous attack script (Bernstein--Lange 25 Dec 2017 17:15:41 -0000) already broke all proposed HK17 parameters. To be more precise, that script takes at most p+1 simple computations, and the largest proposed p was around 2^64 (with a claim of 2^512 pre-quantum security).

The new attack script is practically instantaneous even for p around 2^64. Asymptotically, the number of bit operations in the attack algorithm is (log p)^{1+o(1)} using standard subroutines for basic arithmetic. My impression is that the underlying computer-algebra system already implements these subroutines. The o(1) can be reduced further.

Like the previous attack script, this script focuses on non-degenerate cases. Experimentally, these non-degenerate cases occur almost all the
time: i.e., almost all keys are broken by the script. As before, slightly more work should be able to handle the occasional exceptions.

To use this script, save the following Sage code as break2.sage; copy octonions.py from the HK17 submission to octonions.py in this directory; copy HK17-O.py from the HK17 submission to ref.py in this directory; and run "sage break2.sage". I'm assuming that you have Sage installed. To see the script working for p around 2^64, uncomment the line

```
  modulo=18446744073709551557 #   64 bits
```

in ref.py and then run "sage break2.sage".

Internals: The previous attack script searches for octonions x,y in the commutative ring F_p[oa] such that rA = x ob y, given oa,ob,rA. There are p^8 octonions, but that script applies the following four speedups:

* F_p[oa] is an F_p-vector space of dimension (at most) 2. This means that there are really just 4 variables x0,x1,y0,y1 over F_p.

* The equation rA = x ob y, a system of 8 equations over F_p, is projective. This means that, generically, one can take x1=1, so there are really just 3 variables x0,y0,y1 over F_p.

* The equation splits into rA / x = ob y. This allows a collision search between 1 variable and 2 variables.

* Instead of evaluating ob y, the script checks for each x whether (rA / x) / ob is in F_p[oa].

The new attack script starts from the same 8 equations in 3 variables x0,y0,y1, and applies a straightforward algebraic attack, as suggested by Wang--Malluhi and Li--Liu--Pan--Xie. Specifically, these are linear equations in the monomials

x0y0, x0y1, y0, y1; the script solves for these monomials by linear algebra. It's conceivable that the 8 linear equations have so many redundancies that there isn't a unique solution, but there was always a unique solution in each of a series of several 64-bit experiments.

Presumably an analysis of the failure cases would allow a proof of the failure probability, and, as noted above, an adaptation of the script to handle those cases with slightly more work. But let me emphasize that a fast high-probability attack is already a massive violation of standard security requirements, whether or not the probability is 100%.

---Dan

```
import octonions
import ref
import sys

p = ref.modulo
print ref.message
print ref.times

print '-----'

print "eve observes public parameters and alice's public key:"
print 'p =',p
oa = ref.oa; print 'oa =',oa
ob = ref.ob; print 'ob =',ob
rA = ref.rA; print 'rA =',rA

R.<t0,u0,u1> = GF(p)[]

class oct:
  def __init__(self,*x):
    if len(x) == 1: x = x[0]
    try:
      assert len(x) == 8
      self.c = R(x[0]),R(x[1]),R(x[2]),R(x[3]),R(x[4]),R(x[5]),R(x[6]),R(x[7])
      return
    except:
      self.c = R(x),R(0),R(0),R(0),R(0),R(0),R(0),R(0)

  def __getitem__(self,i):
    return self.c[i]

  def __add__(f,g):
    result = tuple(f[i] + g[i] for i in range(8))
    return oct(result)

  def __sub__(f,g):
    result = tuple(f[i] - g[i] for i in range(8))
    return oct(result)

  def __mul__(x,y):
```

```
    a=x[0]
    b=x[1]
    c=x[2]
    d=x[3]
    e=x[4]
    f=x[5]
    g=x[6]
    h=x[7]
    i=y[0]
    j=y[1]
    k=y[2]
    l=y[3]
    m=y[4]
    n=y[5]
    o=y[6]
    p=y[7]
    t1=(a*i-b*j-c*k-d*l-e*m-f*n-g*o-h*p)
    t2=(a*j+b*i+c*m+d*p-e*k+f*o-g*n-h*l)
    t3=(a*k-b*m+c*i+d*n+e*j-f*l+g*p-h*o)
    t4=(a*l-b*p-c*n+d*i+e*o+f*k-g*m+h*j)
    t5=(a*m+b*k-c*j-d*o+e*i+f*p+g*l-h*n)
    t6=(a*n-b*o+c*l-d*k-e*p+f*i+g*j+h*m)
    t7=(a*o+b*n-c*p+d*m-e*l-f*j+g*i+h*k)
    t8=(a*p+b*l+c*o-d*j+e*n-f*m-g*k+h*i)
    return oct(t1,t2,t3,t4,t5,t6,t7,t8)

  def __repr__(self):
    return '(%s)' % ', '.join('%s' % x.change_ring(ZZ) for x in self)

t = oct(t0) + oct(oa)
u = oct(u0) + oct(u1) * oct(oa)
v = t * oct(ob) * u - oct(rA)

M = matrix(GF(p),8,5)
for i in range(8):
  w = v[i]
  M[i,0] = w[0,0,0]
  M[i,1] = w[0,0,1]
  M[i,2] = w[0,1,0]
  M[i,3] = w[1,0,1]
  M[i,4] = w[1,1,0]
  assert w == M[i,4]*t0*u0+M[i,3]*t0*u1+M[i,2]*u0+M[i,1]*u1+M[i,0]

K = M.right_kernel()
if K.dimension() != 1:
  raise Exception('kernel dimension is not 1, skipping this case for simplicity')

B = K.basis()[0]
if B[0] != 1:
  raise Exception('kernel does not involve constant, skipping this case for simplicity')

x0y0 = B[4]
```

```
x0y1 = B[3]
y0 = B[2]
y1 = B[1]

if y0:
  x0 = x0y0/y0
elif y1:
  x0 = x0y1/y1
else:
  x0 = 0 # key must be 0; probably forces bigger basis anyway

x = oct(list(ti(x0,y0,y1) for ti in t))
y = oct(list(ui(x0,y0,y1) for ui in u))

print "eve's secret key =",x,y

print '-----'

print "now eve looks at bob's ciphertext (DH public key):"
print 'rB =',ref.rB
k = x * oct(ref.rB) * y
print "eve's session key =",k

print '-----'

print 'we now peek at secrets to verify attack worked:'
print "alice's session key =",ref.k1
print "bob's session key =",ref.k2
```

From: Alperin-Sheriff, Jacob (Fed) <jacob.alperin-sheriff@nist.gov>
Sent: Tuesday, January 02, 2018 8:23 AM
To: pqc-forum
Subject: [pqc-forum] FW: OFFICIAL COMMENT: HK17

All:

This also didn't get posted to the forum because it had an attachment (the attachment appears to be identical to this eprint particle by Yanbin Pan et al https://eprint.iacr.org/2017/1259.pdf)

We will try to figure something out about hosting attack code; in the mean time, if possible, please upload it elsewhere and link to it in OFFICIAL COMMENT emails rather than attaching it because in the latter case nobody will see it.

From: Yanbin Pan <panyanbin@amss.ac.cn>
Date: Wednesday, December 27, 2017 at 5:32 AM
To: pqc-comments <pqc-comments@nist.gov>
Cc: pqc-forum <pqc-forum@list.nist.gov>
Subject: OFFICIAL COMMENT: HK17

Dear Dr. Chen, Moody, and Liu,

We group find that the key exchange scheme HK17 is not secure. Any passive adversary can recover the shared key very efficiently.

The key observation is that any octonion (or quaternion) satisfies a quadratic equation, so for any octonion o_A, and any polynomial g(x), we can find a,b such that g(o_A)= a o_A+b.

For HK17, since r_A=f(o_A)^m o_B f(o_A)^n, we can write r_A=(a o_A+b)o_B (c o_A+d) where a,b,c,d are unknowns. By solving a system of linear equations over Z_p, we can find a solution a, b, c d.
Next we can prove that (a o_A+b)r_B (c o_A+d) equals the shared key.

See the attachment for more details.

Best regards,
Yanbin

From: Peter Hecht <qubit101@gmail.com>
Sent: Monday, January 08, 2018 7:19 PM
To: pqc-comments
Cc: pqc-forum@list.nist.gov; Jorge.Kamlofsky@UAI.edu.ar; phecht(DC)
Subject: OFFICIAL COMMENT: HK17

Dear Dan and Tanja, dear all,

Thanks to all for helpful comments.

Despite some pointless objections received, we are convinced that your original idea works as pretended (see Bernstein & Lange, Dec 25.). Later Li et al only confirmed that, but the credit is clearly yours. As a logical consequence, we withdraw our proposal because our modifications are far beyond allowed changes for the first round. We have in mind to block linearizing attacks switching from Moufang loop to GF(2^8) operations, were octonions work now as field members.
More details over our HK17plus protocol could be downloaded (and hopefully commented) at
https://1drv.ms/b/s!ArmCj8o3U1Iyuzd5X8bE9v5sdz57

Best wishes!
Peter