From: Greg Zaverucha <gregz@microsoft.com>
Sent: Monday, August 5, 2019 8:57 PM
To: pqc-forum
Cc: pqc-comments; Daniel Kales
Subject: OFFICIAL COMMENT: MQDSS
Attachments: mqdss_attack.pdf

Hello PQC forum
Recently we found an attack on the proposed parameters for the MQDSS signature scheme. Attached is a short write-up describing the attack. We hope to have a more complete write-up on ePrint in the near future. The MQDSS team has confirmed the attack is valid, and they will send an update to the mailing list.

Greg Zaverucha & Daniel Kales

Forgery Attacks on MQDSSv2.0

Daniel Kales*
Graz University of Technology

Greg Zaverucha
Microsoft Research

August 5, 2019

Abstract

MQDSS is a post-quantum signature scheme, currently a second round candidate in the NIST post-quantum cryptography standardization project. It is built from a 5-pass identification scheme relying on the hardness of the $\mathcal{MQ}$-problem. To transform the 5-pass identification scheme into a signature scheme, a generalization of the Fiat-Shamir transformation is used. In this note, we present a forgery attack on the proposed instances of MQDSS. Concretely, forging a signature for the L1 instance of MQDSS, which should provide 128 bits of security, can be done in $\approx 2^{95}$ hash function calls with high probability. We verify the validity of the attack by implementing it for round reduced versions of MQDSS, and show that we can forge a signature for 40 rounds of MQDSS with $\approx 2^{29}$ hash function calls. Even though a security proof of the scheme exists and we did not find a flaw in it, the proof is not tight enough to rule out these attacks. Our attack does not break the MQDSS design, rather the proposed parameter sets. One mitigation is to increase the number of rounds by a factor of roughly 1.4.

1 Preliminaries

We give a short background on some of the cryptographic building blocks involved in the MQDSS signature scheme. We use the same notation as the MQDSS specification document [CHR+18].

1.1 Canonical $(2n+1)$-pass Identification Schemes

Canonical $(2n+1)$-pass identification schemes are a class of identification schemes which follow a certain message structure. First the prover sends an initial commitment com, then the two parties engage in n rounds, where the verifier sends a challenge ch_i, to which the prover responds with resp_i. We depict a 5-pass identification scheme in Figure 1.

*Work done at Microsoft Research.

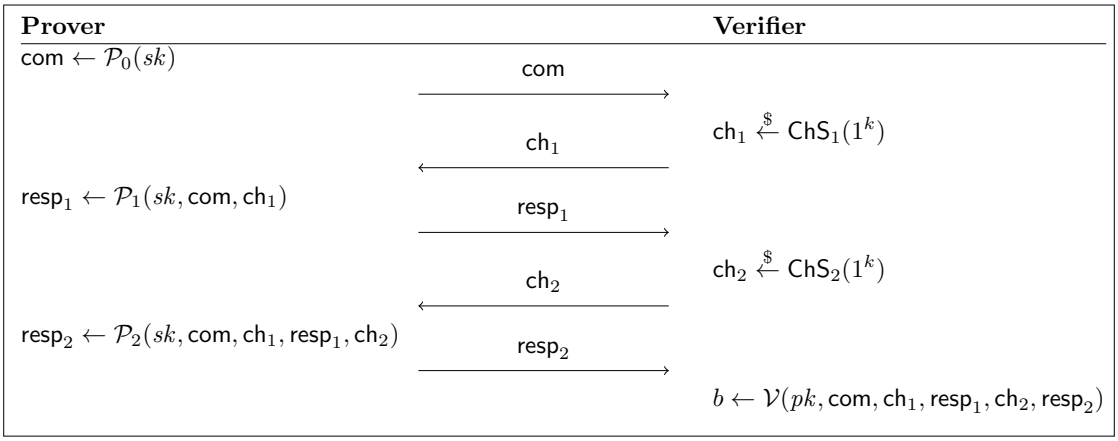

Figure 1: A canonical 5-pass identification scheme.

1.2 Fiat-Shamir Transformation for a class of 5-pass ID schemes

In [CHR+16], Chen et al. give a Fiat-Shamir transformation for a certain class of 5-pass identification schemes. They note that many existing 5-pass identification schemes follow a certain structure, given in Definition 1.

Definition 1 ($q2$-Identification scheme [CHR+16])**.** *Let $k \in \mathbb{N}$. A $q2$-Identification scheme $\mathsf{IDS}(1^k)$ is a canonical 5-pass identification scheme where for the challenge spaces C_1 and C_2 it holds that $|C_1| = q$ and $|C_2| = 2$. Moreover the probability that the commitment com takes a given value is negligible (in k), where the probability is taken over the random choice of the input and the used randomness.*

Definition 2 ($q2$-Extractor [CHR+16])**.** *We say that a $q2$-Identification scheme $\mathsf{IDS}(1^k)$ has a $q2$-extractor if there exists a PPT algorithm $\mathcal{E}$, the extractor, that given a public key pk and four transcripts $\mathsf{trans}^{(i)} = (\mathsf{com}, \mathsf{ch}_1^{(i)}, \mathsf{resp}_1^{(i)}, \mathsf{ch}_2^{(i)}, \mathsf{resp}_2^{(i)}), i \in \{1,2,3,4\}$, with*

$$\mathsf{ch}_1^{(1)} = \mathsf{ch}_1^{(2)} \neq \mathsf{ch}_1^{(3)} = \mathsf{ch}_1^{(4)},$$
$$\mathsf{ch}_2^{(1)} = \mathsf{ch}_2^{(3)} \neq \mathsf{ch}_2^{(2)} = \mathsf{ch}_2^{(4)},$$

which are valid with respect to pk, outputs a matching secret key sk for pk with non-negligible success probability (in k).

The soundness error of an identification scheme, denoted κ, is the probability that the $q2$-extractor fails, and often κ is not sufficiently small to provide cryptographic security. The soundness error can be boosted by running r instances of the identification scheme in parallel. The parameter r is called the number of *rounds* or *parallel repetitions*. Chen et al. [CHR+16] give a proof in the ROM for a Fiat-Shamir transform of $q2$-IDS, if they additionally have a so-called $q2$-extractor (Definition 2).

Construction 1 (Fiat-Shamir transform for $q2$-IDS [CHR+16])**.** *Let $k \in \mathbb{N}$ be the security parameter, $\mathsf{IDS} = (\mathsf{KGen}, \mathcal{P}, \mathcal{V})$ a $q2$-Identification scheme that achieves soundness*

with constant soundness error κ. Select r the number of (parallel) rounds of IDS*, such that $\kappa^r = negl(k)$, and that the challenge spaces of the composition $\mathsf{IDS}^r, C_1^r, C_2^r$ have exponential size in k. Moreover, select cryptographic hash functions $H_1 : \{0,1\}^* \mapsto C_1^r$ and $H_2 : \{0,1\}^* \mapsto C_2^r$. The q2-signature scheme q2-$\mathsf{Dss}(1^k)$ derived from* IDS *is the triplet of algorithms* $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *with:*

- $(sk, pk) \leftarrow \mathsf{KGen}(1^k)$
- $\sigma = (\sigma_0, \sigma_1, \sigma_2) \leftarrow \mathsf{Sign}(sk, m)$ *where* $\sigma_0 = \mathsf{com} \leftarrow \mathcal{P}_0^r(sk), h_1 = H_1(m, \sigma_0), \sigma_1 = \mathsf{resp}_1 \leftarrow \mathcal{P}_1^r(sk, \sigma_0, h_1), h_2 = H_2(m, \sigma_0, h_1, \sigma_1)$ *and* $\sigma_2 = \mathsf{resp}_2 \leftarrow \mathcal{P}_2^r(sk, \sigma_0, h_1, \sigma_1, h_2)$.
- $\mathsf{Vf}(pk, m, \sigma)$ *parses* $\sigma = (\sigma_0, \sigma_1, \sigma_2)$*, computes the values* $h_1 = H_1(m, \sigma_0), h_2 = H_2(m, \sigma_0, h_1, \sigma_1)$ *as above and outputs* $\mathcal{V}^r(pk, \sigma_0, h_1, \sigma_1, h_2, \sigma_2)$.

Theorem 1.1 (EU-CMA security of q2-signature schemes [CHR+16])**.** *Let $k \in \mathbb{N}$,* $\mathsf{IDS}(1^k)$ *be a q*2-IDS *that is honest-verifier zero-knowledge, achieves soundness with constant soundness error κ and has a q2-extractor. Then q*2-$\mathsf{Dss}(1^k)$*, the q2-signature scheme derived applying Construction 1 is existentially unforgeable under adaptive chosen message attacks.*

The proof of Theorem 1.1 is given in [CHR+16]. However, the authors also note that the proof is non-tight. The number of parallel rounds r are chosen according to the soundness error of the underlying IDS, ignoring the potential loss in security that comes from the non-tightness of the proof.

2 Forgery attacks on MQDSS

Chen et al. [CHR+16] give a concrete instantiation – called MQDSS – by applying Construction 1 to the 5-pass identification scheme from [SSH11]. MQDSS is a post-quantum signature scheme submitted to the NIST post-quantum standardization project. For a detailed description of the 5-pass identification scheme we refer to [SSH11, Section 4]. For a complete description of the signature scheme MQDSS we refer to [CHR+18].

MQDSS versions. In August 2018, the MQDSS team updated their specification and recommended parameter sets, due to the original parameters mistakenly being selected for a higher security level. This new parameter sets were called MQDSS v1.1. Additionally, in March 2019 the MQDSS team modified the scheme to include a random string ρ of length 2κ in their commitments. At the time of writing, this is the latest version of MQDSS, MQDSS v2.0. Our attack applies to both, MQDSS v1.1 and v2.0, but in the following, we will use MQDSS v2.0 to be compatible with the most recent reference implementation.

2.1 Description of the Attack on MQDSS

The basic idea of the attack is to split the attacker work between two phases: We try to guess ch_1 for N_1 rounds, and then move on to guess ch_2 for the remaining rounds. For

Algorithm 1 Forge(pk,Msg)

Parse pk as $\mathbf{v}, S_{\mathbf{F}}$
$\mathbf{F} \leftarrow \mathrm{XOF}_{\mathbf{F}}(S_{\mathbf{F}})$
$\mathbf{r}_0^{(1)}, \ldots, \mathbf{r}_0^{(r)}, \mathbf{t}_0^{(1)}, \ldots, \mathbf{t}_0^{(r)}, \mathbf{e}_0^{(1)}, \ldots, \mathbf{e}_0^{(r)} \xleftarrow{\$} \mathbb{F}_q^{n \times 3r}$
$\alpha^* \xleftarrow{\$} \mathbb{F}_q$
$\mathbf{s}^* \xleftarrow{\$} \mathbb{F}_q^n$
for $j \in \{1, \ldots, r\}$ **do**
  $\mathbf{r}_1^{(j)} \leftarrow \mathbf{s}^* - \mathbf{r}_0^{(j)}$
  $\mathbf{t}_1^{(j)} \leftarrow \alpha^* \cdot \mathbf{r}_0^{(j)} - \mathbf{t}_0^{(j)}$
  $\mathbf{e}_1^{(j)} \leftarrow \alpha^* \cdot \mathbf{F}(\mathbf{r}_0^{(j)}) - \mathbf{e}_0^{(j)}$
  $\rho_0^{(j)}, \rho_1^{(j)} \xleftarrow{\$} \{0,1\}^{2\kappa \times 2}$
  $\mathsf{com}_0^{(j)} \leftarrow \mathcal{H}\left(\rho_0^{(j)}, \mathbf{r}_0^{(j)}, \mathbf{t}_0^{(j)}, \mathbf{e}_0^{(j)}\right)$
  $\mathsf{com}_1^{(j)} \leftarrow \mathcal{H}\left(\rho_1^{(j)}, \mathbf{r}_1^{(j)}, \alpha^* \cdot (\mathbf{v} - \mathbf{F}(\mathbf{r}_1^{(j)})) - \mathbf{G}(\mathbf{t}_1^{(j)}, \mathbf{r}_1^{(j)}) - \alpha^* \cdot \mathbf{F}(\mathbf{r}_0^{(j)}) + \mathbf{e}_0^{(j)}\right)$
end for
$\sigma_0 \leftarrow \mathcal{H}(\mathsf{com}_0^{(1)}, \mathsf{com}_1^{(1)}, \ldots, \mathsf{com}_0^{(r)}, \mathsf{com}_1^{(r)})$
repeat
  $R \xleftarrow{\$} \{0,1\}^{2\kappa}$
  $D \leftarrow \mathcal{H}(pk||R||Msg)$
  $\mathsf{ch}_1 \leftarrow H_1(D, \sigma_0)$
  Parse ch_1 as $\mathsf{ch}_1 = \{\alpha^{(1)}, \ldots, \alpha^{(r)}\}, \alpha^{(j)} \in \mathbb{F}_q$
until at least N_1 of $\alpha^{(j)}$ are equal to α^*
repeat
  $\mathsf{guess} \leftarrow \{0,1\}^r$ // in practice, a counter is used to ensure unique hash inputs
  for $j \in \{1, \ldots, r\}$ **do**
    if $\alpha^{(j)} = \alpha^*$ **then**
      $\mathsf{resp}_1^{(j)} \leftarrow (\mathbf{t}_1^{(j)}, \mathbf{e}_1^{(j)})$
    else if bit j of guess is 0 **then**
      $\mathsf{resp}_1^{(j)} \leftarrow \left(\alpha^{(j)} \cdot \mathbf{r}_0^{(j)} - \mathbf{t}_0^{(j)}, \alpha^{(j)} \cdot \mathbf{F}(\mathbf{r}_0^{(j)}) - \mathbf{e}_0^{(j)}\right)$
    else
      $\mathsf{resp}_1^{(j)} \leftarrow \left(\mathbf{t}_1^{(j)}, (\alpha^{(j)} - \alpha^*) \cdot (\mathbf{v} - \mathbf{F}(\mathbf{r}_1^{(j)})) + \alpha^* \cdot \mathbf{F}(\mathbf{r}_0^{(j)}) - \mathbf{e}_0^{(j)}\right)$
    end if
  end for
  $\sigma_1 \leftarrow (\mathsf{resp}_1^{(1)}, \ldots, \mathsf{resp}_1^{(r)})$
  $\mathsf{ch}_2 \leftarrow H_2(D, \sigma_0, \mathsf{ch}_1, \sigma_1)$
until bits of ch_2 agree with guess in positions j where $\alpha^{(j)} \neq \alpha^*$
for $j \in \{1, \ldots, r\}$ **do**
  $\mathsf{resp}_2^{(j)} \leftarrow (\rho_{\mathsf{ch}_2[j]}^{(j)}, \mathbf{r}_{\mathsf{ch}_2[j]}^{(j)})$
end for
$\sigma_2 \leftarrow (\mathsf{resp}_2^{(1)}, \ldots, \mathsf{resp}_2^{(r)})$
return $\sigma = (R, \sigma_0, \sigma_1, \sigma_2)$

many 5-pass identification schemes, including the one used in MQDSS, guessing just one of the two challenges correctly allows the prover to cheat. In the non-interactive version, we leverage the fact that these phases can be repeatedly and separately attacked offline.

In [CHR+16], Chen et al. give a basic strategy for a cheating adversary, that works as follows: The cheater chooses α^* as guess for ch_1 and uses a randomly chosen secret key $\mathbf{s}^*$. He follows the protocol as specified, but computes $\mathbf{r}_1 = \mathbf{s}^* - \mathbf{r}_0$, $\mathbf{t}_1 = \alpha^*\mathbf{r}_0 - \mathbf{t}_0$ and $\mathbf{e}_1 \leftarrow \alpha^* \cdot \mathbf{F}(\mathbf{r}_0) - \mathbf{e}_0$ instead. He also computes the commitment $(\mathsf{com}_0, \mathsf{com}_1)$ as $\mathsf{com}_0 \leftarrow \mathcal{H}(\rho_0, \mathbf{r}_0, \mathbf{t}_0, \mathbf{e}_0)$ and $\mathsf{com}_1 \leftarrow \mathcal{H}(\rho_1, \mathbf{r}_1, \alpha^* \cdot (\mathbf{v} - \mathbf{F}(\mathbf{r}_1)) - \mathbf{G}(\mathbf{t}_1, \mathbf{r}_1) - \alpha^* \cdot \mathbf{F}(\mathbf{r}_0) + \mathbf{e}_0)$. If ch_2 is equal to 0, the recomputed check does not involve the public key $\mathbf{v}$ and will therefore always pass. For $\mathsf{ch}_2 = 1$, the cheater set up the values in a way that the check will still pass if ch_1 was equal to α^*. For our attack, it is important that a bad guess for α^* can be masked by a correct guess of ch_2, without the verifier noticing. This fact allows us to improve the basic attack strategy by trying to guess ch_1 for all parallel repetitions (subsequently fixing any bad guesses in phase 2), not only for a predetermined subset of the repetitions, increasing the success probability to guess N_1 first round challenges correctly from $(\frac{1}{q})^{N_1}$ to $P_1(N_1)$ as given by Equation 1.

Our cheater now has the problem of how to efficiently generate different inputs (still passing verification) to the challenge hash functions H_1 and H_2. For phase 1, this is quite easy, since the signature includes a random salt value R, which is allowed to be chosen freely by the attacker. Therefore an attacker fixes a guess of α^* once, computes the first message σ_0 and then varies R until N_1 of the first challenges agree with α^*. For the second phase we have already fixed R, and can therefore not use the same strategy. However, we can modify the values sent in the second message σ_1 in the following way. While the values of $\mathbf{t}_1$ and $\mathbf{e}_1$ computed as given by the above cheating algorithm are always correct for $\mathsf{ch}_2 = 0$, and fail to verify for $\mathsf{ch}_2 = 1$, we can also come up with different $\mathbf{t}_1$ and $\mathbf{e}_1$ that are correct for $\mathsf{ch}_2 = 1$, but fail for $\mathsf{ch}_2 = 0$. To achieve this we use the same $\mathbf{t}_1$, but compute $\mathbf{e}_1$ such that it corrects the error in com_1, specifically as $\mathbf{e}_1 \leftarrow (\alpha^{(j)} - \alpha^*) \cdot (\mathbf{v} - \mathbf{F}(\mathbf{r}_1)) + \alpha^* \cdot \mathbf{F}(\mathbf{r}_0) - \mathbf{e}_0$. Now our attacker has 2 possible values to send for each second phase round, enabling him to try 2^{r-N_1} different inputs to H_2, and with high probability one of those inputs results in the correct guess for all ch_2 for the remaining $r - N_1$ repetitions. The full attack is given in Algorithm 1.

parameter set	κ	$m = n$	q	r	N_1	$\#\mathcal{H}$	r_{new}
MQDSS-toy	38	48	31	40	11	2^{29}	53
MQDSS-L1	128	48	31	135	41	2^{95}	184
MQDSS-L3	192	64	31	202	61	2^{141}	277
MQDSS-L5	256	88	31	268	82	2^{180}	370

Table 1: Parameter sets for MQDSS instances. r is the number of rounds of MQDSS v2.0, r_{new} is the number of rounds required to resist this attack. (Instance for security level L5 not officially submitted to NIST). N_1 is the number of rounds to attack in the first phase, while $\#\mathcal{H}$ gives an estimate of the required hash function calls for a single forgery.

Alternative for Phase 2. Instead of the adversary trying all different combinations as shown above, he can also fix all but the last repetition, and just vary the responses for this repetition in the following way. Choose a random $\mathbf{t}_1$ and then calculate $\mathbf{e}_1$ as $\mathbf{e}_1 \leftarrow (\alpha^{(j)} - \alpha^*) \cdot (\mathbf{v} - \mathbf{F}(\mathbf{r}_1)) + \alpha^* \cdot \mathbf{F}(\mathbf{r}_0) + \mathbf{G}(\mathbf{t}_0, \mathbf{r}_1) - \mathbf{G}(\mathbf{t}_1, \mathbf{r}_1) - \mathbf{e}_0$. This response is always valid for $\mathsf{ch}_2 = 1$. Due to choosing $\mathbf{t}_1$ at random, we have q^n different possible hash inputs. This method allows us to fix all other repetitions, but requires us to always calculate $\mathbf{G}(\mathbf{t}_1, \mathbf{r}_1)$ instead of being able to cache the output once, like we can do for the other variant. Additionally, this can be used in combination with the other strategy, especially for the case when we exhaust all 2^{r-N_1} possible inputs to H_2, allowing us to continue the attack without having to repeat the first phase.

2.2 Attack Parameters and Mitigation

For the attack, we want to achieve an optimal tradeoff between the work needed for passing the first phase and the work needed for passing the second phase. If we guess N_1 challenges for the first phase correctly, we can answer both possible challenges for these correct guesses in the second phase, only needing to guess the remaining $r - N_1$ second round challenges.

The probability of guessing at least N_1 first round challenges from a challenge space of size $|C_1| = q$ correctly is given by Equation 1.

$$P_1(N_1) = \Pr(\text{guess at least } N_1 \text{ of } r \text{ challenges}) = \sum_{k=N_1}^{r} \left(\frac{1}{q}\right)^k \left(\frac{q-1}{q}\right)^{r-k} \binom{r}{k}. \quad (1)$$

To achieve the best tradeoff in terms of attacker efficiency, we want to minimize the total work for completing both phases. Therefore, the optimal number of rounds to attack in the first phase is given by

$$N_1 = \arg\min_{0 \le N_1' \le r} \left(\frac{1}{P_1(N_1')} + 2^{r-N_1'} \right),$$

i.e., the value of N_1 which minimizes the total cost of the attack. We assume here that both phases are of equal cost, and give some discussion of the cost of the two phases in Section 2.3. A slightly better choice of N_1 is possible by weighting the cost of each phase, based on the costs of a given attack implementation.

We give the numbers for N_1 for different instances in Table 1, together with the estimated number of random oracle calls for a single forgery and the number of parallel repetitions r_{new} that are required so that the expected number of random oracle calls for this attack is at least 2^κ. We stress that we have only briefly investigated mitigation options, and our proposal here should be considered preliminary. After communicating the attack with the MQDSS designers, they have informed us they plan to update the MQDSS specification.

2.3 Practical Verification

To verify the validity of the attack, we implemented it and attacked round-reduced versions of MQDSS. Our MQDSS-toy instance from Table 1 has the same parameters as the instance for the L1 security level, however, we reduced the number of parallel repetitions of the underlying identification scheme from 135 to 40. Based on the formula for the security of MQDSS – $\kappa = (\frac{1}{2} + \frac{1}{2q})^r$ – these 40 rounds should provide about 38 bits of security. The underlying $\mathcal{MQ}$ problem instance is not modified.

Based on our analysis in Section 2.2, we choose the number of rounds to attack in phase 1 to be $N_1 = 11$. The estimated number of random oracle calls is approximately 2^{29}, while for our experiments the average over 10 runs is $2^{27.98}$, all taking between 1 and 12 minutes on a standard desktop PC.

Notes on the implementation. Our implementation also uses a constant amount of memory independent of the security level, making the only real cost producing the inputs to the hash function and executing it, which is very suitable for hardware acceleration. We also provide a more efficient variant of the attack using AVX2 wherever possible and using a Gray code to minimize the changes to the hash input of the second phase. We also observed that even though the first phase has two hash function calls per round compared to the single one for the second phase, the inputs for the second phase hash are much longer, requiring multiple calls to the Keccak permutation (about 1 permutation per 2.25 repetitions). For concrete attack efficiency, this means that attacking 1 or 2 more rounds than given in Table 1 in the first phase is usually faster. We discuss an alternative way to create responses for the second phase in Section 2.1, which can reduce the number of calls to the Keccak permutation to 1, but requires the attacker to evaluate $\mathbf{G}$ instead. Based on our experiments, the cost of $\mathbf{G}$ is about 8 times the cost of the Keccak permutation, which should still result in a faster attack in practice, especially for the larger parameter sets.

References

[CHR+16] Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. From 5-pass MQ-based identification to MQ-based signatures. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 135–165. Springer, Heidelberg, December 2016.

[CHR+18] Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. MQDSS specifications, August 2018. Version 1.1, Available at `mqdss.org/files/MQDSS_Ver1point1.pdf`.

[SSH11] Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. Public-key identification schemes based on multivariate quadratic polynomials. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 706–723. Springer, Heidelberg, August 2011.

From: simona s <simona.samardziska@gmail.com>
Sent: Tuesday, August 6, 2019 4:18 PM
To: pqc-forum; pqc-comments
Subject: Re: [pqc-forum] Re: OFFICIAL COMMENT: MQDSS

Dear all,

We thank Greg and Daniel for their good and insightful work!
Indeed, the MQDSS team has verified the attack and we confirm its validity.
We will take appropriate actions to address the implications of the attack.
As pointed out by Greg and Daniel, the attack asymptotically has exponential complexity.
Hence, it can be mitigated by adjusting the number of rounds by roughly a factor 1.5.
We will post our update on the PQC forum as soon as ready.

We point out that the attack does not invalidate our security proof,
it rather demonstrates that part of the non-tightness which we considered a proof-artifact
is indeed related to a real world attack.
We will work on a more concrete bound for the number of rounds that will
properly reflect the findings of the attack.

The MQDSS team

On Tue, Aug 6, 2019 at 3:14 AM 'daniel.apon' via pqc-forum <pqc-forum@list.nist.gov> wrote:

> Thanks, Greg Z / Daniel K,
>
> We'll review this immediately.
>
> --Daniel A
>
> On Monday, August 5, 2019 at 8:56:39 PM UTC-4, Greg Zaverucha wrote:
>
> > Hello PQC forum
> >
> > Recently we found an attack on the proposed parameters for the MQDSS signature scheme. Attached is a short write-up describing the attack. We hope to have a more complete write-up on ePrint in the near future. The MQDSS team has confirmed the attack is valid, and they will send an update to the mailing list.
> >
> > Greg Zaverucha & Daniel Kales