Ming-Shing Chen
Andreas Hülsing
Joost Rijneveld
Simona Samardjiska
Peter Schwabe

MQDSS specifications

Version 2.0 March 2019

Introduction

This document is a detailed specification of the design and security arguments of the digital signature scheme MQDSS. It is divided in two main parts:

- **Part I Backbone results** - contains:
  - an analysis of the hardness of the underlying hard problem with respect to both classical and quantum algorithms - Chapter 2,
  - a description of the underlying Identification scheme - Chapter 3,
  - a description and proof of security of the underlying construction - Chapter 5.
- **Part II MQDSS Specifications** - contains:
  - a detailed description of MQDSS - Chapter 7 and Chapter 9,
  - proposed and additional parameter sets - Chapter 8,
  - security analysis of MQDSS - Chapter 10,
  - justification of the design choices - Chapter 11,
  - a detailed performance analysis of the reference implementation using the proposed parameter sets - Chapter 12.
  - a discussion on the security vs performance tradeoffs - Chapter 13,
  - a summary of the strengths and weaknesses - Chapter 14, and
  - a short description of the optimized AVX2 implementation - Chapter 15.

New in Version 2.0

Changes were made to the commitment functions. In particular the commitment functions now take an additional argument - a random string of length $2k$. The reason for this change is that with this additional input it can be shown that the commitment function used in MQDSS is computationally hiding - a propery needed to show the EU-CMA security of MQDSS. We use a recent result from [45].

We have updated the algorithms for key generation, signing and verification accordingly to reflect this change (see Chapter 7 and Chapter 9 for details). We have also updated the security analysis (see Chapter 10 and Appendix A). Note that the size of the signature for all parameter sets is bigger than in Version 1.1, but still significantly smaller than the signature sizes from the NIST submission, i.e., Version 1.0.

Furthermore, a more accurate analysis for the best classical attacks against the $\mathcal{MQ}$ problem was done, and the estimated classical complexity is now given in terms of gates.

New in Version 1.1

The parameter sets are updated to reflect the changes made to the number of rounds r (see Chapter 5 for definition of the parameter, and Chapter 8 for the calculation of the parameter). In particular, the necessary number of rounds is half of the one given in Version 1.0. This has been corrected in Version 1.1.

A consequence of the update is that for all security levels, the signature size is approximately half of the one presented in Version 1.0.

Note that further, as a result of the new way the parameters are determined, there is also a reduction in size of the public and the secret keys (See Chapter 8 for details, and compare to the same chapter in Version 1.0).

Contents

Part II MQDSS Specifications

Part I

Backbone Results
- Underlying Construction and Security Arguments -

1 Preliminaries

1.1 Notations and Conventions

Let $A(\cdot, \cdot, \dots)$ be a randomized algorithm. We write $y \leftarrow A(x_1, x_2, \dots)$ for the output of the algorithm on input $x_1, x_2, \dots$. The same notation is used for the output of a function. If S is a set, then $s \leftarrow_R S$ denotes that s is drawn uniformly at random from S.

Furthermore, let $\mathbb{F}_q$ denote the finite field of order q. We use boldface letters $\mathbf{u}$ to denote vectors over a finite field, i.e. $\mathbf{u} \in \mathbb{F}_q^n$, for some positive integer $n \in \mathbb{N}$. We call a function $\mathbb{F}_q^n \to \mathbb{F}_q^m$ a vectorial function.

1.2 Security Notions and Definitions

In the following we provide basic security related definitions used throughout these specifications.

A function μ is called negligible (in k) if for every positive polynomial p, and sufficiently large k it holds that $\mu(k) < 1/p(k)$. For better readability we sometimes denote negligible functions by $\mathrm{negl}(k)$.

We say that two distribution ensembles $\{X_k\}_{k\in\mathbb{N}}$ and $\{Y_k\}_{k\in\mathbb{N}}$ indexed by a security parameter k are computationally indistinguishable if for any non-uniform probabilistic polynomial time algorithm $\mathcal{A}$

$$|\mathsf{Pr}\left[1 \leftarrow \mathcal{A}\left(X_k\right)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A}\left(Y_k\right)\right]| = \mathrm{negl}(k)\,.$$

1.2.1 Digital Signatures

This specification describes a construction of digital-signature schemes. These are defined as follows.

Definition 1.1 (Digital signature scheme). *A digital-signature scheme with security parameter k, denoted $\mathsf{Dss}(1^k)$ is a triplet of polynomial-time algorithms $\mathsf{Dss} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ defined as follows:*

- *The key-generation algorithm* KGen *is a probabilistic algorithm that outputs a key pair* $(\mathsf{sk}, \mathsf{pk})$.
- *The signing algorithm* Sign *is a possibly probabilistic algorithm that on input a secret key* sk *and a message M outputs a signature σ.*

- *The verification algorithm* Vf *is a deterministic algorithm that on input a public key* pk*, a message* M *and a signature* σ *outputs a bit* b*, where* $b = 1$ *indicates that the signature is accepted and* $b = 0$ *indicates a reject.*

We write Dss instead of $\mathsf{Dss}(1^k)$, whenever the security parameter k is clear from context or irrelevant. For correctness of a Dss, we require that for all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}()$, all messages M and all signatures $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, M)$, we get $\mathsf{Vf}(\mathsf{pk}, M, \sigma) = 1$, i.e., that correctly generated signatures are accepted.

Existential Unforgeability under Adaptive Chosen Message Attacks.
The standard security notion for digital signature schemes is existential unforgeability under adaptive chosen message attacks (EU-CMA) [35], defined as follows.

```
Experiment $\mathsf{Exp}^{\mathsf{eu\text{-}cma}}_{\mathsf{Dss}(1^k)}(\mathcal{A})$
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}()$
    $(M^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk})$
    Let $\{(M_i)\}_1^{Q_s}$ be the queries to $\mathsf{Sign}(\mathsf{sk}, \cdot)$.
    Return 1 iff $\mathsf{Vf}(\mathsf{pk}, M^\star, \sigma^\star) = 1$ and $M^\star \notin \{M_i\}_1^{Q_s}$.
```

For the success probability of an adversary $\mathcal{A}$ in the above experiment we write

$$\mathrm{Succ}^{\mathsf{eu\text{-}cma}}_{\mathsf{Dss}(1^k)}(\mathcal{A}) = \Pr\left[\mathsf{Exp}^{\mathsf{eu\text{-}cma}}_{\mathsf{Dss}(1^k)}(\mathcal{A}) = 1\right].$$

A signature scheme is called EU-CMA-secure if any PPT algorithm $\mathcal{A}$ has only negligible success probability in the $\mathsf{Exp}^{\mathsf{eu\text{-}cma}}_{\mathsf{Dss}(1^k)}(\mathcal{A})$ experiment. More formally, we have the following definition.

Definition 1.2 (EU-CMA security). *Let* $k \in \mathbb{N}$ *and* Dss *a digital signature scheme with security parameter* k*. We call* Dss *existentially unforgeable under chosen message attacks or EU-CMA-secure if for all* $Q_s, t = \mathrm{poly}(k)$ *the success probability of any PPT algorithm* $\mathcal{A}$ *(the adversary) running in time* $\leq t$*, making at most* Q_s *queries to* Sign *in the* $\mathsf{Exp}^{\mathsf{eu\text{-}cma}}_{\mathsf{Dss}(1^k)}(\mathcal{A})$ *experiment, is negligible in* k*:*

$$\mathrm{Succ}^{\mathsf{eu\text{-}cma}}_{\mathsf{Dss}(1^k)}(\mathcal{A}) = \mathrm{negl}(k).$$

In the security proof of our signature scheme, we will also make use of the weaker notion of security against key-only attacks (KOA). The difference from EU-CMA security is that the adversary is given no access to the signing oracle, i.e., $Q_s = 0$. More formally, we define the following experiment.

```
Experiment $\mathsf{Exp}^{\mathsf{koa}}_{\mathsf{Dss}(1^k)}(\mathcal{A})$
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}()$
    $(M^\star, \sigma^\star) \leftarrow \mathcal{A}(\mathsf{pk})$
    Return 1 iff $\mathsf{Vf}(\mathsf{pk}, M^\star, \sigma^\star) = 1$.
```

Definition 1.3 (KOA security). *Let* $k \in \mathbb{N}$ *and* Dss *a digital signature scheme with security parameter* k*. We call* Dss *secure under key only attacks or KOA-secure if for all* $t = \mathrm{poly}(k)$ *the success probability of any PPT adversary* $\mathcal{A}$ *running in time* $\leq t$ *in the* $\mathsf{Exp}^{\mathsf{koa}}_{\mathsf{Dss}(1^k)}(\mathcal{A})$ *experiment, is negligible in* k*:*

$$\mathrm{Succ}^{\mathsf{koa}}_{\mathsf{Dss}(1^k)}(\mathcal{A}) = \mathrm{negl}(k),$$

where $\mathrm{Succ}^{\mathsf{koa}}_{\mathsf{Dss}(1^k)}(\mathcal{A}) = \Pr\left[\mathsf{Exp}^{\mathsf{koa}}_{\mathsf{Dss}(1^k)}(\mathcal{A}) = 1\right].$

1.2.2 Identification Schemes

An identification scheme (IDS) is a protocol that allows a prover $\mathcal{P}$ to prove its identity to a verifier $\mathcal{V}$. More formally:

Definition 1.4 (Identification scheme). *An identification scheme with security parameter k, denoted $\mathsf{IDS}(1^k)$, is a triplet of PPT algorithms $\mathsf{IDS} = (\mathsf{KGen}, \mathcal{P}, \mathcal{V})$ such that:*

- *the key generation algorithm KGen outputs a key pair $(\mathsf{sk}, \mathsf{pk})$.*
- *$\mathcal{P}$ and $\mathcal{V}$ are interactive algorithms, executing a common protocol. The prover $\mathcal{P}$ takes as input a secret key sk and the verifier $\mathcal{V}$ takes as input a public key pk. At the conclusion of the protocol, $\mathcal{V}$ outputs a bit b with $b = 1$ indicating "accept" and $b = 0$ indicating "reject".*

We write IDS instead of $IDS(1^k)$, if the security parameter k is clear from context or irrelevant. For correctness of an IDS, we require that for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}()$ we have

$$\mathsf{Pr}\left[\langle \mathcal{P}(\mathsf{sk}), \mathcal{V}(\mathsf{pk})\rangle = 1\right] = 1,$$

where $\langle \mathcal{P}(\mathsf{sk}), \mathcal{V}(\mathsf{pk})\rangle$ refers to the common execution of the protocol between $\mathcal{P}$ with input sk and $\mathcal{V}$ on input pk. In this case we say that the IDS is perfectly correct.

For the following definitions we need the notion of a transcript. A transcript of an execution of an identification scheme IDS refers to all the messages exchanged between $\mathcal{P}$ and $\mathcal{V}$ and is denoted by $\mathsf{trans}(\langle \mathcal{P}(\mathsf{sk}), \mathcal{V}(\mathsf{pk})\rangle)$.

We will focus on canonical $2n+1$-pass IDS, where the prover and the verifier exchange $2n+1$ messages, n challenges and n replies. These IDS are defined as follows.

Definition 1.5 (Canonical $2n+1$-pass identification schemes). *Consider $\mathsf{IDS} = (\mathsf{KGen}, \mathcal{P}, \mathcal{V})$, a $2n+1$-pass identification scheme with n challenge spaces $\mathsf{C}_1, \ldots, \mathsf{C}_n$. We call IDS a canonical $2n+1$-pass identification scheme if the prover can be split into $n+1$ subroutines $\mathcal{P} = (\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_n)$ and the verifier into $n+1$ subroutines $\mathcal{V} = (\mathsf{ChS}_1, \ldots, \mathsf{ChS}_n, \mathsf{Vf})$ such that:*

- *$\mathcal{P}_0(\mathsf{sk})$ computes the initial commitment com sent as the first message and a state state fed forward to $\mathcal{P}_1$.*
- *ChS_1, computes the first challenge message $\mathsf{ch}_1 \leftarrow_R \mathsf{C}_1$, sampling at random from the challenge space C_1.*
- *$\mathcal{P}_1(\mathsf{state}, \mathsf{ch}_1)$, computes the first response resp_1 of the prover (and updates the state state) given access to the state and the first challenge.*
- *For every $i \in \{2, \ldots, n\}$*
  - *ChS_i, computes the i-th challenge message $\mathsf{ch}_i \leftarrow_R \mathsf{C}_i$.*
  - *$\mathcal{P}_i(\mathsf{state}, \mathsf{ch}_i)$, computes the i-th response resp_i of the prover given access to the state and the i-th challenge.*
- *$\mathsf{Vf}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \ldots, \mathsf{ch}_n, \mathsf{resp}_n)$, upon access to the public key and the whole transcript outputs $\mathcal{V}$'s final decision.*

Note that the state forwarded among the prover algorithms can contain all inputs to previous prover algorithms if they are needed later. We also assume that the verifier keeps all sent and received messages to feed them to Vf.

Our construction uses the special case of canonical 5-pass IDS (where $n = 2$). On the other hand, standard choice in the literature for building signatures is the special case

$n = 1$. For comparison, we will use both in these specifications, and for completeness and clarity we provide figures of both. Figure 1.1 describes a canonical 3-pass IDS, and Figure 1.2 a canonical 5-pass IDS.

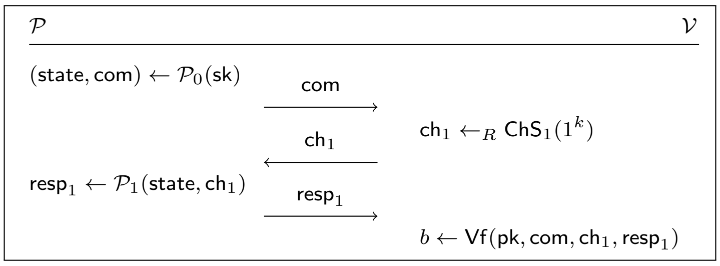
$\mathcal{P}$
$\mathcal{V}$
$(\mathsf{state}, \mathsf{com}) \leftarrow \mathcal{P}_0(\mathsf{sk})$
com
$\mathsf{ch}_1 \leftarrow_R \mathsf{ChS}_1(1^k)$
ch_1
$\mathsf{resp}_1 \leftarrow \mathcal{P}_1(\mathsf{state}, \mathsf{ch}_1)$
resp_1
$b \leftarrow \mathsf{Vf}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1)$

Fig. 1.1: Canonical 3-pass IDS

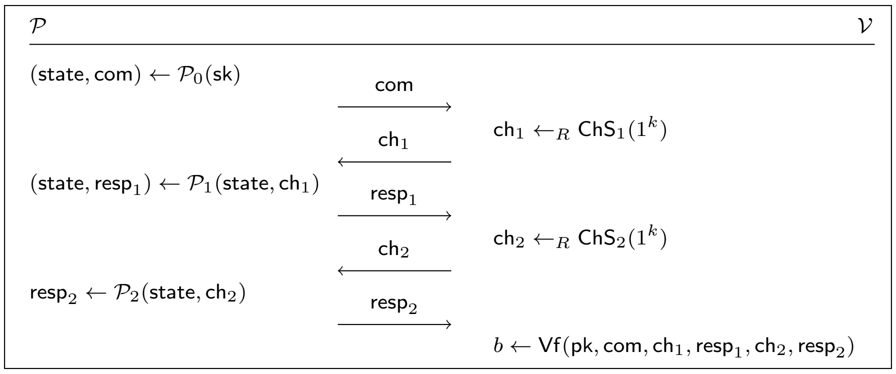
$\mathcal{P}$
$\mathcal{V}$
$(\mathsf{state}, \mathsf{com}) \leftarrow \mathcal{P}_0(\mathsf{sk})$
com
$\mathsf{ch}_1 \leftarrow_R \mathsf{ChS}_1(1^k)$
ch_1
$(\mathsf{state}, \mathsf{resp}_1) \leftarrow \mathcal{P}_1(\mathsf{state}, \mathsf{ch}_1)$
resp_1
$\mathsf{ch}_2 \leftarrow_R \mathsf{ChS}_2(1^k)$
ch_2
$\mathsf{resp}_2 \leftarrow \mathcal{P}_2(\mathsf{state}, \mathsf{ch}_2)$
resp_2
$b \leftarrow \mathsf{Vf}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}_2, \mathsf{resp}_2)$

Fig. 1.2: Canonical 5-pass IDS

Furthermore, we will consider a particular type of canonical 5-pass IDS where the size of the two challenge spaces is restricted to q and 2.

Definition 1.6 (**$q2$-Identification scheme**). *A $q2$-Identification scheme* IDS *is a canonical 5-pass identification scheme where for the challenge spaces* C_1 *and* C_2 *it holds that* $|\mathsf{C}_1| = q$ *and* $|\mathsf{C}_2| = 2$.

Security against Impersonation under Passive Attack.
The standard security notion for identification schemes is security against impersonation. Here, the goal of the adversary - the impersonator $\mathcal{I}$, is to impersonate the prover in an interaction with an honest verifier without the knowledge of the secret key. When talking about passive attacks, the impersonator, besides the public key, might have access to polynomially many valid interactions between the prover and the verifier (via eavesdropping for example), i.e., access to a transcript oracle $\mathsf{Trans}(\mathsf{pk}, \mathsf{sk}, \cdot)$ that outputs valid transcripts of honest executions.

For a canonical $2n + 1$-pass IDS we consider the following experiment:

```
Experiment $\mathsf{Exp}^{\mathsf{imp\text{-}pa}}_{\mathsf{IDS}(1^k)}(\mathcal{I})$
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}()$
    $(\mathsf{state}, \mathsf{com}) \leftarrow \mathcal{I}^{\mathsf{Trans}(\mathsf{pk},\mathsf{sk},\cdot)}(\mathsf{pk})$
    For every $i \in \{1, \ldots, n\}$
        $\mathsf{ch}_i \leftarrow_R \mathsf{ChS}_i(1^k)$
        $(\mathsf{state}, \mathsf{resp}_i) \leftarrow \mathcal{I}^{\mathsf{Trans}(\mathsf{pk},\mathsf{sk},\cdot)}(\mathsf{pk})$
    Return 1 iff $\mathsf{Vf}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \ldots, \mathsf{ch}_n, \mathsf{resp}_n) = 1$.
```

For the success probability of the impersonator $\mathcal{I}$ in the above experiment we write

$$\mathrm{Succ}^{\mathsf{imp\text{-}pa}}_{\mathsf{IDS}(1^k)}(\mathcal{I}) = \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{imp\text{-}pa}}_{\mathsf{IDS}(1^k)}(\mathcal{I}) = 1\right].$$

Definition 1.7 (IMP-PA security). *Let $k \in \mathbb{N}$ and IDS a canonical $2n+1$ identification scheme with security parameter k. We say IDS is secure against impersonation under passive attacks or IMP-PA-secure if for all $Q_t, t = \mathrm{poly}(k)$ the success probability of any PPT impersonator $\mathcal{I}$ running in time $\leq t$, making at most Q_t queries to Trans in the $\mathsf{Exp}^{\mathit{imp\text{-}pa}}_{\mathsf{IDS}(1^k)}(\mathcal{I})$ experiment, is negligible in k:*

$$\mathrm{Succ}^{\mathit{imp\text{-}pa}}_{\mathsf{IDS}(1^k)}(\mathcal{I}) = \mathrm{negl}(k).$$

Security Properties of Identification Schemes.
The properties of identification schemes interesting in our context are those that provide passive security. We next give the necessary definitions.

First of all, it must be hard for any cryptographic scheme to derive a valid secret key given a public key. To formally capture this intuition, we need to define what valid means. For this we define the notion of a key relation.

Definition 1.8 (Key relation). *Let IDS be an identification scheme and R some relation. We say IDS has key relation R if*

$$\forall(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}() : (\mathsf{pk}, \mathsf{sk}) \in R$$

Now that we have defined what valid means, we can define key-one-wayness.

Definition 1.9 (Key-One-Wayness). *Let $k \in \mathbb{N}$ be the security parameter, $\mathsf{IDS}(1^k)$ be an identification scheme with key relation R. We call IDS key-one-way (KOW) (with respect to key relation R) if for all polynomial time algorithms $\mathcal{A}$,*

$$\mathrm{Succ}^{pq-kow}_{\mathsf{IDS}(1^k)}(\mathcal{A}) = \mathsf{Pr}\left[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(), \mathsf{sk}' \leftarrow \mathcal{A}(\mathsf{pk}) : (\mathsf{pk}, \mathsf{sk}') \in R\right] = \mathrm{negl}(k)$$

Definition 1.10 (Soundness (with soundness error κ)). *Let $k \in \mathbb{N}$, $\mathsf{IDS}(1^k) = (\mathsf{KGen}, \mathcal{P}, \mathcal{V})$ an identification scheme with security parameter k. We say that IDS is sound, with soundness error κ, if for every PPT algorithm $\mathcal{A}$ (the adversary),*

$$\mathsf{Pr}\left[\begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}() \\ \left\langle \mathcal{A}(1^k, \mathsf{pk}), \mathcal{V}(\mathsf{pk})\right\rangle = 1\end{array}\right] \leq \kappa + \mathrm{negl}(k).$$

Definition 1.11 ((computational) Honest-verifier zero-knowledge). *Let $k \in \mathbb{N}$, $\mathsf{IDS}(1^k) = (\mathsf{KGen}, \mathcal{P}, \mathcal{V})$ an identification scheme with security parameter k. We say that IDS is computational honest-verifier zero-knowledge (HVZK) if there exists a probabilistic polynomial time algorithm $\mathcal{S}$, called the simulator, such that for any polynomial time algorithm $\mathcal{A}$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}()$:*

$$\begin{aligned}&\mathrm{Succ}^{pq-hvzk}_{\mathsf{IDS}(1^k)}(\mathcal{A}) = \\ &\quad |\mathsf{Pr}\left[1 \leftarrow \mathcal{A}\left(\mathsf{sk}, \mathsf{pk}, \mathsf{trans}(\langle \mathcal{P}(\mathsf{sk}), \mathcal{V}(\mathsf{pk})\rangle)\right)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A}\left(\mathsf{sk}, \mathsf{pk}, \mathcal{S}(\mathsf{pk})\right)\right]| = \mathrm{negl}(k).\end{aligned}$$

Definition 1.12 (Special soundness). *Let* $\mathsf{IDS}(1^k)$ *be a* 3*-pass Identification scheme with key relation* R *and* $\mathcal{A}$ *a polynomial time algorithm that upon input of security parameter* 1^k *and an* $\mathsf{IDS}(1^k)$ *public key* pk *outputs, with non-negligible probability, four valid transcripts with respect to* pk*:*

$$\begin{aligned} \mathsf{trans} &= (\mathsf{com},\ \mathsf{ch}_1,\ \mathsf{resp}_1), \\ \mathsf{trans}' &= (\mathsf{com},\ \mathsf{ch}'_1,\ \mathsf{resp}'_1), \end{aligned} \tag{1.1}$$

where $\mathsf{ch}_1 \neq \mathsf{ch}'_1$*.*

We say that $\mathsf{IDS}(1^k)$ *is special sound if there exists a polynomial time algorithm* $\mathcal{K}_{IDS}$*, the extractor, that, given a public key* pk *and access to* $\mathcal{A}$*, outputs a secret key* sk *such that* $(\mathsf{pk}, \mathsf{sk}) \in R$ *with non-negligible success probability in* k*.*

To prove security of our signature scheme, we will make use of the existence of so called $q2$-extractor which is a variant of special soundness. This is combined with a notion of key-one-wayness to later be able to argue about security.

Definition 1.13 ($q2$-Extractor). *Let* $\mathsf{IDS}(1^k)$ *be a* $q2$*-Identification scheme with key relation* R *and* $\mathcal{A}$ *a polynomial time algorithm that upon input of security parameter* 1^k *and an* $\mathsf{IDS}(1^k)$ *public key* pk *outputs, with non-negligible probability, four valid transcripts with respect to* pk*:*

$$\begin{aligned} \mathsf{trans}^{(1)} &= (\mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}_2, \mathsf{resp}_2), & \mathsf{trans}^{(3)} &= (\mathsf{com}, \mathsf{ch}'_1, \mathsf{resp}'_1, \mathsf{ch}_2, \mathsf{resp}_2), \\ \mathsf{trans}^{(2)} &= (\mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}'_2, \mathsf{resp}'_2), & \mathsf{trans}^{(4)} &= (\mathsf{com}, \mathsf{ch}'_1, \mathsf{resp}'_1, \mathsf{ch}'_2, \mathsf{resp}'_2). \end{aligned} \tag{1.2}$$

where $\mathsf{ch}_1 \neq \mathsf{ch}'_1$ *and* $\mathsf{ch}_2 \neq \mathsf{ch}'_2$*.*

We say that $\mathsf{IDS}(1^k)$ *has a* $q2$-Extractor *if there exists a polynomial time algorithm* $\mathcal{K}_{IDS}$*, the extractor, that, given a public key* pk *and access to* $\mathcal{A}$*, outputs a secret key* sk *such that* $(\mathsf{pk}, \mathsf{sk}) \in R$ *with non-negligible success probability in* k*.*

Security Properties of Commitments.
The security of identification schemes relies on the properties of the underlying commitment scheme. The goal of "committing" to a certain value is twofold: It should not be feasible for anyone to discover this value before the prover opens the commitment, but also, it should not be feasible for the prover to open the commitment in multiple ways. These two properties are known as hiding and binding. They come in different flavors - perfect, statistical and computational, depending on what "feasible" means.

For our purposes, the weakest version of computational hiding and binding will suffice. These are formally defined as follows.

Definition 1.14 (Computationally hiding commitments). *Let* $k \in \mathbb{N}$*,* $\mathsf{Com}(1^k)$ *a commitment scheme with security parameter* k*. We say that* Com *is computationally hiding if for any two messages* M, M' *and random string* ρ*, for any polynomial time algorithm* $\mathcal{A}$*:*

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A}\left(\mathsf{Com}(\rho, M)\right)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A}\left(\mathsf{Com}(\rho, M')\right)\right]\right| = \mathrm{negl}(k)\,.$$

Definition 1.15 (Computationally binding commitments). *Let* $k \in \mathbb{N}$*,* $\mathsf{Com}(1^k)$ *a commitment scheme with security parameter* k*. We say that* Com *is computationally binding if for any polynomial time algorithm* $\mathcal{A}$

$$\mathsf{Pr}\left[\mathsf{Com}(\rho, M) = \mathsf{Com}(\rho', M') \wedge M \neq M' : (\rho, M, \rho', M') \leftarrow \mathcal{A}(1^k)\right] = \mathrm{negl}(k)\,.$$

2

The $\mathcal{MQ}$ Problem

2.1 Multivariate Quadratic ($\mathcal{MQ}$) Functions and the $\mathcal{MQ}$ Problem

Let $m, n, q \in \mathbb{N}$, $\mathbf{x} = (x_1, \ldots, x_n)$ and let $\mathcal{MQ}(n, m, \mathbb{F}_q)$ denote the family of vectorial functions $\mathbf{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ of degree 2 over $\mathbb{F}_q$:

$$\begin{aligned}\mathcal{MQ}(n, m, \mathbb{F}_q) = \{\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))|& \\ f_s(\mathbf{x}) = \sum_{i,j} a_{i,j}^{(s)} x_i x_j + \sum_i b_i^{(s)} x_i, s \in \{1, \ldots, m\}\}.&\end{aligned}$$

We will refer to $\mathbf{F} \in \mathcal{MQ}(n, m, \mathbb{F}_q)$ as multivariate quadratic ($\mathcal{MQ}$) function. Given $\mathbf{v} \in \mathbb{F}_q^m$ we will refer to $\mathbf{F}(\mathbf{x}) = \mathbf{v}$ as system of m quadratic equations in n variables.

We will omit m, n, q whenever they are clear from the context.

Definition 2.1. *Let $\mathbf{F} \in \mathcal{MQ}(n, m, \mathbb{F}_q)$. The function $\boldsymbol{G}(\mathbf{x}, \mathbf{y}) = \boldsymbol{F}(\mathbf{x} + \mathbf{y}) - \boldsymbol{F}(\mathbf{x}) - \boldsymbol{F}(\mathbf{y})$ is called the polar form of the function $\boldsymbol{F}$.*

It is not hard to verify that the polar form is bilinear, i.e., for every $\mathbf{a}_1, \mathbf{a}_2, \mathbf{b} \in \mathbb{F}_q^n$ it holds

$$\begin{aligned}\mathbf{G}(\mathbf{a}_1 + \mathbf{a}_2, \mathbf{b}) &= \mathbf{G}(\mathbf{a}_1, \mathbf{b}) + \mathbf{G}(\mathbf{a}_2, \mathbf{b}) \text{ and} \\ \mathbf{G}(\mathbf{b}, \mathbf{a}_1 + \mathbf{a}_2) &= \mathbf{G}(\mathbf{b}, \mathbf{a}_1) + \mathbf{G}(\mathbf{b}, \mathbf{a}_2).\end{aligned}$$

Definition 2.2 ($\mathcal{MQ}$ relation). *The $\mathcal{MQ}$ relation is the binary relation defined as: $R_{\mathcal{MQ}(m,n,q)} \subseteq (\mathcal{MQ}(n, m, \mathbb{F}_q) \times \mathbb{F}_q^m) \times \mathbb{F}_q^n : ((\mathbf{F}, \mathbf{v}), \mathbf{s}) \in R_{\mathcal{MQ}(m,n,q)}$ iff $\mathbf{F}(\mathbf{s}) = \mathbf{v}$.*

We relate the following problem to the family $\mathcal{MQ}(n, m, \mathbb{F}_q)$ of $\mathcal{MQ}$ functions:

Definition 2.3 ($\mathcal{MQ}$ problem (search version)). *Let $m, n, q \in \mathbb{N}$. An instance $\mathcal{MQ}(\mathbf{F}, \mathbf{v})$ of the $\mathcal{MQ}$ (search) problem is defined as:*

Given $\mathbf{F} \in \mathcal{MQ}(n, m, \mathbb{F}_q), \mathbf{v} \in \mathbb{F}_q^m$ find, if any, $\mathbf{s} \in \mathbb{F}_q^n$ such that

$$((\mathbf{F}, \mathbf{v}), \mathbf{s}) \in R_{\mathcal{MQ}(m,n,q)}.$$

The decisional version of the $\mathcal{MQ}$ problem is known to be NP-complete [33][1]. It is widely believed that the $\mathcal{MQ}$ problem is intractable even for quantum computers in the average case. We formalize the intractability of the $\mathcal{MQ}$ problem through the following.

[1] Note that the $\mathcal{MQ}$ problem is a special case of the more general problem of solving a system of equations over a finite field of degree $deg \geq 2$, known as $\mathcal{P}o\mathcal{SS}o$. The decisional version of the $\mathcal{P}o\mathcal{SS}o$ problem is NP-complete [33].

Assumption 2.4 ($\mathcal{MQ}$ assumption) *Let $m, n, q \in \mathbb{N}$, $\mathbf{F} \leftarrow_R \mathcal{MQ}(n, m, \mathbb{F}_q)$ and $\mathbf{s} \leftarrow_R \mathbb{F}_q^n$. For every polynomial time quantum algorithm $\mathcal{A}$ given $\mathbf{F}$ and $\mathbf{v} = \mathbf{F}(\mathbf{s})$ it is computationally hard to find a solution $\mathbf{s}'$ to the $\mathcal{MQ}(\mathbf{F}, \mathbf{v})$ problem. More formally,*

$$\Pr\left[\big((\mathbf{F},\mathbf{v}),\mathbf{s}'\big) \in R_{\mathcal{MQ}(m,n,q)} : \begin{array}{l} \mathbf{F} \leftarrow_R \mathcal{MQ}(n,m,\mathbb{F}_q) \\ \mathbf{s} \leftarrow_R \mathbb{F}_q^n \\ ((\mathbf{F},\mathbf{v}),\mathbf{s}) \in R_{\mathcal{MQ}(m,n,q)} \\ \mathbf{s}' \leftarrow \mathcal{A}(1^k,\mathbf{F},\mathbf{v}) \end{array} \right] = \operatorname{negl}(k)\,.$$

2.2 Classical Algorithms for Solving the $\mathcal{MQ}$ Problem

The difficulty of solving the $\mathcal{MQ}$ problem is strongly dependent on the ratio between the number of variables n and number of equations m. It is known that when $m \geqslant n(n+1)/2$ (overdetermined systems) and when $n \geqslant m(m+1)$ (underdetermined systems) the $\mathcal{MQ}$ problem is solvable in polynomial time.

The first case is simply a result of replacing all monomials by a new variable, and solving a linear system in $n(n+1)/2$ variables and at least as many equations. The second case was solved by Kipnis, Patarin and Goubin [42] and later [56] Thomae and Wolf showed that the complexity gradually increases to exponential when $m \approx n$. Indeed, the most interesting case is when $m = n$: Adding more equations gives away more information about the system; On the other hand, if there are more variables, we can simply fix the excess of them, and end up with a system of the same number of variables and equations.

In the rest of this section we will assume that $m \geqslant n$, but also that $m = \mathcal{O}(n)$. For this range of parameters, the state of the art algorithms employ algebraic techniques that analyze the properties of the ideal generated by the given polynomials. The most important are the algorithms from the F4/F5 family [27, 28, 5, 12], and the variants of the XL algorithm [21, 24, 62, 61]. Although different in description, the two families bear many similarities, which results in similar complexity [63]. Therefore, in our analysis we will not consider the algorithms from the XL family.

In the Boolean case, today's state of the art algorithms BooleanSolve [7] and FXL [61] provide improvement over exhaustive search with an asymptotic complexity of $\Theta(2^{0.792n})$ and $\Theta(2^{0.875n})$ for $m = n$, respectively. Practically, the improvement is visible for polynomials with more than 200 variables. A very recent algorithm, the Crossbred algorithm [38] over $\mathbb{F}_2$, is likely to further improve the asymptotic complexity, as the authors report that it passes the exhaustive search barrier already for 37 Boolean variables.

Interestingly, the current best known algorithms, BooleanSolve [7], FXL [61, 62], the Crossbred algorithm [38] and the Hybrid approach [12] all combine algebraic techniques with exhaustive search. This immediately allows for improvement in their quantum version using Grover's quantum search algorithm [37], provided the cost of implementing them on a quantum computer does not diminish the gain from Grover. These algorithms will be subject to our interest in the rest of the section. Their quantum version and the speed up from using Grover's algorithm will be discussed in Section 2.3.

For comparison reasons, in our analysis we will also consider exhaustive search performed through fast enumeration techniques [13]. We will not consider a probabilistic method recently proposed by Lokshtanov et al. [46]. Although it is provably faster than exhaustive search, the improvement in the case of odd characteristic fields is not comparable to the best algebraic methods. Furthermore, it has not been studied enough and has not been implemented (to the best of our knowledge), so even in the Boolean case where the asymptotic complexity is $\mathcal{O}(2^{0.8765n})$ it is not clear for what values of n this algorithm outperforms exhaustive search.

In the rest of this section, let $\mathbf{F} = (f_1, \dots, f_m), f_i \in \mathbb{F}_q[x_1, \dots, x_n]$. Without loss of generality, the equation system that we want to solve is $\mathbf{F}(\mathbf{x}) = \mathbf{0}$.

2.2.1 Exhaustive search

A natural and simple way of obtaining a solution of the given system is to try out all possible values $\mathbf{x} \in \mathbb{F}_q^n$ until the system is satisfied. A naïve implementation would require $2\binom{n}{2}$ additions and multiplications for a single polynomial, and m times more for the entire system, amounting to a complexity of $\mathcal{O}(mn^2q^n)$ field operations. However, in [13], Bouillaguet *et al.* introduced a technique for fast enumeration in $\mathbb{F}_2$ that needs only $\log_2(n)2^{n+2}$ Boolean operations. The technique uses Gray codes enumeration and partial derivatives of the polynomials. Although [13] considers only the Boolean case, the technique can be extended to larger fields by using q-ary Gray codes. So, for simplicity we will assume that fast enumeration can be performed in $\log_q(n)q^n$ operations over a field of size q.

2.2.2 The HybridF5 algorithm

Currently, the standard algorithms for solving generic instances of the $\mathcal{MQ}$ problem are the algorithms for computing a Gröbner basis of the ideal generated by the set of $\mathcal{MQ}$ polynomials. The idea was first introduced by Buchberger [15] and later further developed by Lazard [44] who established the link between computing the Gröbner basis and performing Gaussian elimination on the Macaulay matrices (of degree up to a sufficiently large integer D) of the given polynomials.[2] The algorithm was improved several times by Buchberger himself in order to reduce the number of unnecessary reductions to 0 during the Gröbner basis computation. A significant improvement was done in the variant proposed by Faugère [27] known as the F4-algorithm. The main improvement comes from the introduced strategy to reduce all critical pairs of minimal degree at once (instead of one by one) using the Macaulay matrix and sparse matrix algebra techniques. Later, Faugère completely removed the reductions to zero for semi-regular sequences in the improved F5-algorithm [28, 8, 6].

The semi-regularity assumption is crucial in this algorithm (as it will be in the other algebraic methods we consider). Informally (which is enough for our purposes), a sequence of polynomials $(f_1, \dots, f_m), m \geq n$ is semi-regular if the only relations (dependencies) among the polynomials are the trivial ones generated by $f_i f_j - f_j f_i = 0$. Note that the regularity assumption is a very plausible one for randomly generated polynomials, and has been experimentally supported (see for example [7]). We will also assume that the instances generated in our signature scheme are semi-regular.

The main complexity in the F5 algorithm (and also in the BooleanSolve and the Crossbread algorithm) comes from performing Gauss elimination on the Macaulay matrix $Mac_D(\mathbf{F})$ of degree D (the matrix whose rows are formed by the coefficients of monomials

[2] In essence, it can be considered as a generalization of Gaussian elimination for nonlinear polynomials. One important distinction is that, unlike in Gaussian elimination, in Gröbner basis algorithms the order in which the variables are eliminated and generally, the ordering of the monomials, is very important (see [28, 8] for example).

uf_i of maximal degree D). The degree D should be big enough so that a Gröbner basis of the ideal generated by the polynomials can be obtained by row-reducing the Macaulay matrix. The smallest such D is called the degree of regularity D_{reg}, and for semi-regular systems it is given by $D_{reg}(n,m) = 1 + deg(HS_q(t))$, where

$$HS_q(t) = \left[\frac{(1-t^2)^m}{(1-t)^n}\right]_+, \text{for } q > 2, \text{ and } HS_2(t) = \left[\frac{(1+t)^n}{(1+t^2)^m}\right]_+,$$

and the $_+$ subscript denotes that the series has been truncated before the first non-positive coefficient.

Now, for the case of $q > 2$ the Macaulay matrix $Mac_D(\mathbf{F})$ has $\binom{n+D-1}{D}$ columns and $m\binom{n+D-3}{D-2}$ rows, so computing the row-echelon form requires $\Theta(m\binom{n+D-3}{D-2}\binom{n+D-1}{D}^{\omega-1})$ operations, where $2 \leqslant \omega \leqslant 3$ is the linear algebra constant. The computation is repeated for every $D \in \{2, \dots, D_{reg}\}$. In the case of $q = 2$, the logic is the same, except that now we use plain combinations (instead of combinations with repetition as for $q > 2$). In total, the complexity of the F5 algorithm in field operations is:

$$\begin{aligned} C_{F5}(q,m,n) &= \Theta\left(m\sum_{D=2}^{D_{reg}}\binom{n+D-3}{D-2}\binom{n+D-1}{D}^{\omega-1}\right), \text{ for } q > 2, \text{ and} \\ C_{F5}(2,m,n) &= \Theta\left(m\sum_{D=2}^{D_{reg}}\binom{n}{D-2}\binom{n}{D}^{\omega-1}\right) \end{aligned} \tag{2.1}$$

More compactly, the complexity of the F5 algorithm in field operations [6] is:

$$\begin{aligned} C_{F5}(q,m,n) &= \mathcal{O}\left(\binom{n+D_{reg}(n,m)-1}{D_{reg}(n,m)}^{\omega}\right), \text{ for } q > 2, \text{ and} \\ C_{F5}(2,m,n) &= \mathcal{O}\left(\binom{n}{D_{reg}(n,m)}^{\omega}\right) \end{aligned}$$

The value of the linear algebra constant ω depends on the algorithm used, and it ranges from $\omega = 3$ for naïve Gauss elimination down to $\omega = 2.376$ for Coppersmith-Winograd algorithm [19], and even further to $\omega < 2.373$ due to improvements by Vassilevska-Williams [60]. However these algorithms are extremely complex and with a huge constant factor to be actually useful in practice. For cryptanalysis purposes, the best we can hope for is $\omega = \log_2(7)$, obtained using Strassen algorithm [54].

The Hybrid approach introduced by Bettale *et al.*[12], tries to reduce the complexity of F5 by introducing a trade-off between brute-forcing and the F5 algorithm for smaller $\mathcal{MQ}$ instances. Namely, the algorithms first fixes $n-k$ variables, so the reduction is now performed on $Mac_{D_{reg}}(\tilde{\mathbf{F}})$, where $\tilde{\mathbf{F}} = (\tilde{f}_1, \dots, \tilde{f}_m)$ and $\tilde{f}_i(x_1, \dots, x_k) = f_i(x_1, \dots, x_k, a_{k+1}, \dots, a_n)$, for every $(a_{k+1}, \dots, a_n) \in \mathbb{F}_2^{n-k}$. The value of k is chosen such that the overall complexity is minimized. In total the complexity of the Hybrid approach for solving systems of n equations in n variables over $\mathbb{F}_q$ is

$$C_{Hyb}(n,k) = Guess(q, n-k) \cdot C_{F5}(q,k,n), \tag{2.2}$$

where $Guess(q, n-k) = \mathcal{O}(\log(n-k)q^{n-k})$ is the cost of the exhaustive search over all q^{n-k} possibilities, including partially evaluating $n-k$ variables in field operations, and $C_{F5}(k,n)$ is given in (2.1).

Note that the technique of fixing variables had already been used in the XL algorithm [21], and this version is known as FXL [61, 62].

2.2.3 The BooleanSolve algorithm

In the case of $\mathbb{F}_2$, the BooleanSolve algorithm [7] performs better than the Hybrid approach. Similar to the Hybrid approach, it requires a semi-regularity assumption on the $\mathcal{MQ}$ instance. Also as in the Hybrid approach, it first fixes some optimal amount $n-k$ of the variables and then performs some tests on the smaller instance.

Then, the problem of finding a solution is basically reduced to testing the consistency of a related linear system

$$\mathbf{u} \cdot Mac_{D_{reg}}(\tilde{\mathbf{F}}) = (0, \ldots, 0, 1) \tag{2.3}$$

where $Mac_{D_{reg}}(\tilde{\mathbf{F}})$ is defined in the previous paragraph. If the system (2.3) is consistent, then the original system does not have a solution. This allows for pruning of all the inconsistent branches corresponding to some $\mathbf{a} \in \mathbb{F}_2^{n-k}$. A simple exhaustive search is then performed on the remaining branches. It can be shown that the running time of the algorithm is dominated by the first part of the algorithm (this holds true even in the quantum version of the algorithm, although in the quantum case the difference is not as big, as a consequence of the reduced complexity of the first part). Therefore, for simplicity, we omit the exhaustive search on the remaining branches from our analysis. The complexity of the BooleanSolve algorithm is given by

$$C_{Bool}(n,k) = Guess(2, n-k) \cdot C_{cons}(Mac_{D_{reg}}(\tilde{\mathbf{F}})), \tag{2.4}$$

where $Guess(2,k)$ is defined the same as in the Hybrid approach, and

$$C_{cons}(Mac_{D_{reg}}(\tilde{\mathbf{F}})) = \Theta(N^2 \log^2 N \log\log N), \quad N = \sum_{i=0}^{D_{reg}(k,n)} \binom{k}{i}$$

is the complexity of testing consistency of the system (2.3), using the sparse linear algebra algorithm from [34].

2.2.4 The Crossbread algorithm

Recently, Joux and Vitse [38] proposed a new algebraic method for solving quadratic systems over $\mathbb{F}_2$ called the Crossbred algorithm. Although originally only $\mathbb{F}_2$ was considered, the algorithm works the same for any field, so we will assume an arbitrary field $\mathbb{F}_q$. We will also assume that the given system is semi-regular.

The main idea of this approach is to first perform some operations on the Macaulay matrix of degree $D \geqslant D_{reg}(k,n)$ of the given system, and only afterwards to fix variables. Again, as in the previous algorithms, k is a suitably chosen optimization parameter such that the overall complexity is minimized. Furthermore, let $d \leqslant D$ be a small integer and $deg_k u$ denote the degree of the monomial u in the first k variables. Let $Mac_{D,d}^{(k)}(\mathbf{F})$ be the submatrix of $Mac_D(\mathbf{F})$ consisting of the rows indexed by uf_i, where $deg_k u \geqslant d-1$, and let $M_{D,d}^{(k)}(\mathbf{F})$ be the submatrix of $Mac_{D,d}^{(k)}(\mathbf{F})$ consisting of the columns indexed by u, where $deg_k u > d$.

The algorithm works as follows: In the first part, we try to find enough linearly independent elements v_i (in particular for $q > 2$ at least $\sum_{i=0}^{d} \binom{k+i-1}{i}$ including the original m when $d > 1$) in the kernel of $M_{D,d}^{(k)}(\mathbf{F})$, that are not in the kernel of $Mac_{D,d}^{(k)}(\mathbf{F})$. Next we find the set of polynomials corresponding to $v_i Mac_{D,d}^{(k)}(\mathbf{F})$. These polynomials, (possibly together with the original when $d > 1$) form a new system $\mathbf{P}$ that will be of interest in the second part of the algorithm.

In this part, for each $(a_{k+1}, \dots, a_n) \in \mathbb{F}_q^{n-k}$ we form the system $\tilde{\mathbf{P}}(x_1, \dots, x_k) = \mathbf{P}(x_1, \dots, x_k, a_{k+1}, \dots, a_n)$. It is crucial to observe that $\tilde{\mathbf{P}}$ is of degree d and the system contains $\sum_{i=0}^{d} \binom{k+i-1}{i}$ equations when $q > 2$ ($\sum_{i=0}^{d} \binom{k}{i}$ when $q = 2$), which means it is possible to solve it easily by linearization, i.e. by considering each monomial as new variable, and solving the resulting linear system.

The advantage here comes from using sparse linear algebra algorithms on $Mac_D(\mathbf{F})$ for the first part and dense linear algebra only on the smaller matrix in the second part. Note that, as long as the number of the remaining k variables is small, the sparse linear algebra part takes much less time, since in this case $D_{reg}(k, n)$ is also small. It turns out that actually it is more efficient to work with a Mac_D, with $D > D_{reg}(k, n)$, but not too large so that the cost of the first part becomes significant. The complexity thus, is dominated by enumeration of $n-k$ variables in a system of n variables of degree D over $\mathbb{F}_q$ $q > 2$, and checking whether the obtained system has a valid solution. In total, under the condition that:

$$Dim(Ker(M_{D,d}^{(k)}(\mathbf{F})) \setminus Ker(Mac_{D,d}^{(k)}(\mathbf{F}))) \geqslant \sum_{i=0}^{d} \binom{k+i-1}{i}, \tag{2.5}$$

the complexity of the Crossbread algorithm for $q > 2$ is given by[3]

$$C_{Cross}(n, k, d) = Sparse(M_{D,d}^{(k)}(\mathbf{F})) + Guess(q, n-k) \cdot \binom{k+d-1}{d}^{\omega}, \tag{2.6}$$

where $Guess(q, k)$ is defined the same as in the Hybrid approach, and $Sparse(M_{D,d}^{(k)}(\mathbf{F})) = \mathcal{O}(\binom{n+D-1}{D}^2)$ is the complexity for finding the kernel vectors using for example the block Lanczos algorithm [47] or the block Wiedemann algorithm [20] for sparse matrices (or their improvements). Note that an external specialization of variables is also possible, but we have verified that this does not bring any improvement in the number of operations. However it is useful for parallelization of the algorithm. [4]

At the end of this section, we provide the complexity of the described algorithms for solving $\mathcal{MQ}$ instances for various fields $\mathbb{F}_q$ and different values of $m = n$ that are interesting for practical use. Table 2.1 summarizes the Boolean case, and Table 2.2 lists the values for some common choices of $q > 2$. The cost of the algorithms is given in gate count[5] as suggested in the NIST call for proposals [49]. In this transformation from field operations to gate count, we have assumed that a field multiplication costs $2\lceil \log_2 q \rceil^2$ gates and a field addition costs $\lceil \log_2 q \rceil$ gates.

[3] The case of $q = 2$ is similar except that the system is of size $\approx \binom{k}{d}$.

[4] The preprint [38] does not contain a complexity analysis of the Crossbread algorithm, nor are all the choices in the algorithm described. What is written here is our interpretation of the algorithm.

[5] In Version 1 of these specifications we used field operations for estimating the classical complexity of the algorithms.

	CrossBread ($d=1$)		BooleanSolve		HybridF5		FastEnum
n	k	Gates	k	Gates	k	Gates	Gates
128	28	2^{119}	40	2^{137}	16	2^{139}	2^{134}
144	32	2^{132}	52	2^{151}	17	2^{155}	2^{150}
160	30	2^{149}	55	2^{165}	18	2^{170}	2^{166}
192	31	2^{180}	80	2^{193}	35	2^{200}	2^{199}
224	32	2^{212}	96	2^{220}	38	2^{230}	2^{231}
256	33	2^{243}	102	2^{248}	55	2^{261}	2^{263}
296	34	2^{282}	139	2^{282}	59	2^{298}	2^{303}

Table 2.1: Comparison of the time complexity of the Crossbread algorithm [38], the BooleanSolve algorithm [7], the Hybrid Approach [12] and exhaustive search through fast enumeration [13] in terms of field operations for $\mathbb{F}_2$. The parameter k denotes the number of remaining variables in the specialization process in each of the algorithms respectively.

		HybridF5		CrossBread ($d=1$)		FastEnum
q	n	k	Gates	k	Gates	Gates
4	80	32	2^{157}	20	2^{139}	2^{167}
4	96	35	2^{185}	21	2^{169}	2^{199}
4	112	44	2^{212}	22	2^{200}	2^{231}
4	128	53	2^{240}	23	2^{230}	2^{263}
4	144	51	2^{266}	24	2^{260}	2^{295}
4	160	60	2^{293}	25	2^{290}	2^{327}
16	48	36	2^{148}	17	2^{144}	2^{200}
16	64	41	2^{191}	19	2^{201}	2^{264}
16	72	49	2^{211}	20	2^{229}	2^{296}
16	80	52	2^{233}	20	2^{261}	2^{328}
16	96	66	2^{273}	21	2^{321}	2^{392}
31	40	32	2^{136}	17	2^{135}	2^{206}
31	48	39	2^{160}	17	2^{174}	2^{246}
31	64	49	2^{206}	19	2^{244}	2^{325}
31	72	53	2^{230}	20	2^{279}	2^{365}
31	80	62	2^{253}	20	2^{319}	2^{404}
31	88	71	2^{275}	20	2^{359}	2^{444}
31	96	72	2^{298}	21	2^{394}	2^{484}
32	48	39	2^{161}	17	2^{176}	2^{248}
32	64	52	2^{207}	19	2^{246}	2^{328}
32	72	53	2^{231}	20	2^{282}	2^{368}
32	80	62	2^{253}	20	2^{322}	2^{408}
32	88	71	2^{276}	20	2^{362}	2^{448}
32	96	72	2^{299}	21	2^{397}	2^{488}
64	40	32	2^{145}	17	2^{159}	2^{248}
64	64	52	2^{220}	19	2^{292}	2^{392}
128	40	37	2^{150}	17	2^{183}	2^{288}
128	64	59	2^{230}	19	2	2^{456}
256	40	37	2^{153}	17	2^{206}	2^{328}

Table 2.2: Comparison of the time complexity of the the Hybrid Approach [12], the Crossbread algorithm [38], and exhaustive search through fast enumeration [13] in terms of field operations for common choices of the field $\mathbb{F}_q$, $q > 2$. The parameter k denotes the number of remaining variables in the specialization process in each of the algorithms respectively.

2.3 Using Grover's Algorithm for Solving the $\mathcal{MQ}$ Problem

In this section we will investigate the cost of the quantum versions of the known classical algorithms that we described in the previous Section. To the best of our knowledge, there are no dedicated quantum algorithms for solving the $\mathcal{MQ}$ problem. We are only aware of the work of Westerbaan and Schwabe [59], who investigate the cost of exhaustive search using Grover's algorithm against the $\mathcal{MQ}$ problem. In our paper [16], where we first introduce MQDSS, we briefly analyze the gain of applying Grover on the Hybrid approach.

Here, we will use a more accurate metric, and following NIST's recommendations [49] we will express the cost of the algorithms in terms of number of fault-tolerant quantum gates and quantum circuit depth.

2.3.1 Finite Field Arithmetic on Quantum Computers

Fault-Tolerant quantum gates.
In quantum computing, similarly as in classical computing, there is a need for fixed small universal set of instructions that can be used to express any type of reversible quantum operation. Furthermore, such universal sets need to have fault-tolerant implementations to reduce pilling up of noise and thus errors in quantum computation. Recent work [4, 3] has identified "Clifford+T" as the standard universal fault-tolerant gate set. It is the set of gates generated by $= \{H, CNOT, T\}$ where,

$$\begin{aligned} H &: |x\rangle \mapsto \frac{|0\rangle + (-1)^x|1\rangle}{\sqrt{2}}, \\ CNOT &: |x\rangle|y\rangle \mapsto |x\rangle|x \oplus y\rangle, \\ T &: |x\rangle \mapsto e^{\frac{i\pi x}{4}}|x\rangle. \end{aligned}$$

We will also need the Toffoli gate:

$$Toffoli : |x\rangle|y\rangle|z\rangle \mapsto |x\rangle|y\rangle|z \oplus xy\rangle.$$

which is also a common gate in designing circuits. Many implementations of the Toffoli gate using Clifford+T gate are known, depending on whether the goal is to minimize the number of ancilla qubits used, the gate count or circuit depth. In our evaluation we have chosen a balanced metric, assuming sufficient number of ancilla qubits. In other words, we are interested in implementations that minimize at the same time the T-count and T-depth (of T gates only) but more importantly, the overall gate count and depth including Clifford gates. Thus we will use the implementation from Amy *et al.*[4] that requires 7 T-gates and 8 Clifford gates, and has T-depth 4, and overall depth 8.

In what follows, we will use the same metric to evaluate larger quantum circuits.

Cost of finite field addition and multiplication.
The algorithms we are interested in, are all performed over finite fields $\mathbb{F}(2^s)$ or $\mathbb{F}_p$ for p prime. Therefore we need an estimate for the cost of the arithmetic operations over these fields. We use the results from [9, 18, 39]:

- Addition over $\mathbb{F}(2^s)$ can be implemented using s parallel $CNOT$ (Clifford) gates (so the overall depth is 1).

- Multiplication over $\mathbb{F}(2^s)$ (using Karatsuba's algorithm) can be implemented using $7s^{\log_2(3)}$ T-gates and $10s^{\log_2(3)} - 2s$ Clifford gates, with T-depth of $4s$ and overall depth of $9s$
- Addition over $\mathbb{F}(p)$ can be implemented using approximately $180\log_2(p)$ Clifford gates and $140\log_2(p)$ T-gates with the same depth.
- Multiplication over $\mathbb{F}(p)$ can be implemented using approximately $2\cdot180\log_2^2(p)$ Clifford gates and $2\cdot140\log_2^2(p)$ T-gates with T-depth of $2\cdot140\log_2(p)$ and overall depth of $2\cdot320\log_2(p)$.

2.3.2 Grover's Quantum Search Algorithm

Grover's algorithm [37] searches for an item in an unordered list of size $N = 2^k$ that satisfies a certain condition given in a form of a quantum black-box function $f : \{0,1\}^k \to \{0,1\}$ and realized as a unitary circuit $U_f : |x\rangle|y\rangle \mapsto |x\rangle|x \oplus f(y)\rangle$ - the "oracle". If the condition is satisfied for an item x_0, then $f(x_0) = 1$, otherwise $f(x_0) = 0$. The algorithm consists of applying an optimal number of times the operator $G = U_f\left((H^{\otimes k}(2|0\rangle\langle 0| - 1_{2^k})H^{\otimes k}) \otimes 1_2\right)$ on a state $|\psi\rangle \otimes |\phi\rangle$ where the first register has been prepared in an equal superposition of all $|x\rangle$, i.e., $|\psi\rangle = \frac{1}{\sqrt{2^k}}\sum_{x\in\{0,1\}^k}|x\rangle$, and $\phi = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ (see Figure 2.1). The operator G needs to be iteratively repeated $\left\lfloor \frac{\pi}{4}\sqrt{N/M} \right\rfloor$ times, where M is the number of items that satisfy the condition f [14]. In this case, if $M \ll N$, the algorithm fails with negligible probability $\leqslant M/N$. Note that, even if the number of solutions M is unknown, a slight modification of the algorithm from [14], again guarantees that a solution will be found with overwhelming probability after $\left\lfloor \frac{9}{4}\sqrt{N/M} \right\rfloor$ Grover iterations. In the next subsection, we will elaborate on the number of solutions of the $\mathcal{MQ}$ problem.

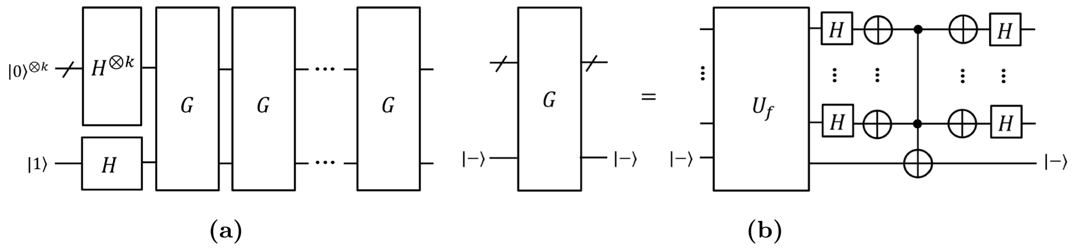

Fig. 2.1: Quantum circuit that implements Grover's algorithm for a search space of size $N = 2^k$. Figure from [36]. (a) The full algorithm where G is the Grover iterate that represents one round of the algorithm. (b) One round of Grover's algorithm (detailed view of the operator G).

From the above, and assuming we know the number of solutions, the cost of Grover's algorithm can be expressed as:

$$Cost(Grover) = \left\lfloor \frac{\pi}{4}\sqrt{2^k/M} \right\rfloor \cdot (Cost(U_f) + Cost(U_s)) \tag{2.7}$$

where $U_s = 2|s\rangle\langle s| - 1_{2^k}$ is the Grover diffusion operator. Here $Cost$ can be any metric of choice, such as quantum gate count or quantum circuit depth.

In [36] it was calculated that the diffusion operator U_s can be implemented as a k-fold $CNOT$ gate which requires $8k-24$ Toffoli gates, which easily translates to Clifford+T gates (see previous Section).

In the next subsection we will consider several different instantiations of the function U_f leading to a solution for the $\mathcal{MQ}$ problem $\mathbf{F}(\mathbf{x}) = \mathbf{0}$. The oracle U_f can be any of the following:

- The $\mathcal{MQ}$ oracle $U_{\mathcal{MQ}}$: $U_{\mathcal{MQ}}(\mathbf{a}) = 1$ for $\mathbf{a} \in \mathbb{F}_q^n$ if $\mathbf{F}(\mathbf{a}) = \mathbf{0}$ (cf. Subsection (2.2.1));
- The BooleanSolve oracle U_{Bool}: $U_{Bool}(\mathbf{a}) = 1$ for $\mathbf{a} \in \mathbb{F}_2^{n-k}$ if the system (2.3) is inconsistent and the second part of the BooleanSolve algorithm on $\tilde{\mathbf{F}}$ outputs $\mathbf{b} \in \mathbb{F}_q^k$ such that $\tilde{\mathbf{F}}(\mathbf{b}) = \mathbf{0}$ (cf. Subsection (2.2.3));
- The Hybrid F5 oracle U_{HybF5}: $U_{HybF5}(\mathbf{a}) = 1$ for $\mathbf{a} \in \mathbb{F}_q^{n-k}$ if the F5 algorithm on $\tilde{\mathbf{F}}$ outputs $\mathbf{b} \in \mathbb{F}_q^k$ such that $\tilde{\mathbf{F}}(\mathbf{b}) = \mathbf{0}$ (cf. Subsection (2.2.2));
- The Crossbread oracle U_{Cross}: $U_{Cross}(\mathbf{a}) = 1$ for $\mathbf{a} \in \mathbb{F}_q^{n-k}$ if the Crossbread algorithm on $\tilde{\mathbf{P}}$ outputs a solution $\mathbf{b} \in \mathbb{F}_q^k$ such that $\tilde{\mathbf{P}}(\mathbf{b}) = \mathbf{0}$ (cf. Subsection (2.2.4)).

2.3.3 Resource Estimates of Grover Enhanced Quantum Algorithms for Solving the $\mathcal{MQ}$ Problem

On the number of solutions of the $\mathcal{MQ}$ problem.
As already mentioned, our proposal uses randomly generated instances $\mathbf{F} \in \mathcal{MQ}(n, m, \mathbb{F}_q)$, where the number of polynomials m is the same as the number of variables n. The goal of the adversary will be to find one solution of a system $\mathbf{F}(\mathbf{x}) = \mathbf{v}$, for a given public value $\mathbf{v} \in \mathbb{F}^n$. While our key generation mechanism guarantees that this system will have at least one solution, we don't know the exact number of solutions which is an important parameter in Grover's algorithm. In the previous section we saw that it is possible to overcome this problem by adapting Grover's algorithm to such a setting. But, we can actually argue that there is no need for that, and that it is safe to assume in our analysis that the number of solutions is $M = 1$. Indeed, in [31], it was shown that the number of solutions of a system of n equations in n variables follows the Poisson distribution with parameter $\lambda = 1$ (the expected value is 1), i.e. the probability that the system has exactly M solutions is $\frac{1}{eM!}$. Furthermore, the probability that there are more than M solutions can be estimated as the tail probability of a Poisson distribution which is negligible in M. This means that with overwhelming probability, the number of solutions is very small, and we can simply run Grover first assuming $M = 1$, then $M = 2$ and so on, until the algorithm succeeds. In particular, since we know that the system has at least one solution, the probability that it is the only solution is $\frac{1}{e-1} \approx 0.58$, and that there are at most 2 solutions $\frac{5}{4(e-1)} \approx 0.73$. Hence, the adversary has a good chance to succeed already in the first two runs, and the probability quickly rises with each additional run. In our analysis, we will assume that it is enough to run Grover only for $M = 1$ (as a lower bound of the cost of the algorithm).

The $\mathcal{MQ}$ oracle.
In [59], Westerbaan and Schwabe constructed two oracles for evaluation of $\mathcal{MQ}$ polynomials over $\mathbb{F}_2$ and estimated the cost of Grover's algorithm using these oracles. Here we will adapt their estimates for the case of any field $\mathbb{F}_q$. As our metrics is mainly circuit size and depth (and not number of qubits) we will focus on their approach for the first oracle.

Their second oracle uses approximately half the number of qubits of their first oracle (with a small overhead), but double the circuit size.

Following [59] we estimate that the $\mathcal{MQ}$ oracle $U_{\mathcal{MQ}}$ over $\mathbb{F}_q$ requires approximately $4n^2m$ field multiplications and as many field additions. The total depth required for the multiplications is approximately $4n$, as is for the additions. Using the formulas:

$$\begin{aligned} Gates(\mathcal{MQ}Grover) &= \left\lfloor \frac{\pi}{4} 2^{\lfloor \log_2 q \rfloor \frac{n}{2}} \right\rfloor \cdot (Gates(U_{\mathcal{MQ}}) + Gates(U_s)), \\ Depth(\mathcal{MQ}Grover) &= \left\lfloor \frac{\pi}{4} 2^{\lfloor \log_2 q \rfloor \frac{n}{2}} \right\rfloor \cdot (Depth(U_{\mathcal{MQ}}) + Depth(U_s)), \end{aligned} \tag{2.8}$$

and the results from Subsections 2.3.1 and 2.3.2 we obtain an estimate of the cost of Exhaustive search with Grover's algorithm. The results are summarized in Table 2.3.

		Gates		Depth	
q	n	T	Clifford	T	Total
2	128	$2^{89.46}$	$2^{89.82}$	$2^{76.54}$	$2^{77.63}$
2	192	$2^{123.21}$	$2^{123.58}$	$2^{109.13}$	$2^{110.23}$
2	224	$2^{139.88}$	$2^{140.24}$	$2^{125.36}$	$2^{126.45}$
2	256	$2^{156.46}$	$2^{156.82}$	$2^{141.55}$	$2^{142.64}$
4	72	$2^{96.55}$	$2^{96.97}$	$2^{85.02}$	$2^{86.15}$
4	96	$2^{121.80}$	$2^{122.21}$	$2^{109.44}$	$2^{110.57}$
4	112	$2^{138.47}$	$2^{138.88}$	$2^{125.66}$	$2^{126.79}$
4	128	$2^{155.04}$	$2^{155.46}$	$2^{141.85}$	$2^{142.98}$
16	32	$2^{86.63}$	$2^{87.08}$	$2^{77.21}$	$2^{78.38}$
16	40	$2^{103.60}$	$2^{104.04}$	$2^{93.53}$	$2^{94.70}$
16	48	$2^{120.38}$	$2^{120.83}$	$2^{109.80}$	$2^{110.96}$
16	64	$2^{153.63}$	$2^{154.08}$	$2^{142.22}$	$2^{143.38}$
31	24	$2^{87.77}$	$2^{88.13}$	$2^{78.60}$	$2^{79.80}$
31	32	$2^{108.83}$	$2^{109.19}$	$2^{98.84}$	$2^{100.03}$
31	40	$2^{129.61}$	$2^{129.97}$	$2^{118.98}$	$2^{120.17}$
31	48	$2^{150.22}$	$2^{150.58}$	$2^{139.06}$	$2^{140.25}$
31	56	$2^{170.70}$	$2^{171.06}$	$2^{159.09}$	$2^{160.29}$
32	32	$2^{103.14}$	$2^{103.60}$	$2^{93.66}$	$2^{94.84}$
32	40	$2^{124.11}$	$2^{124.56}$	$2^{113.99}$	$2^{115.16}$
32	48	$2^{144.89}$	$2^{145.35}$	$2^{134.25}$	$2^{135.43}$
32	56	$2^{165.56}$	$2^{166.02}$	$2^{154.47}$	$2^{155.65}$
64	24	$2^{94.31}$	$2^{94.78}$	$2^{85.62}$	$2^{86.80}$
64	32	$2^{119.56}$	$2^{120.02}$	$2^{110.04}$	$2^{111.22}$
64	40	$2^{144.52}$	$2^{144.99}$	$2^{134.36}$	$2^{135.54}$
64	48	$2^{169.31}$	$2^{169.77}$	$2^{158.62}$	$2^{159.81}$
128	24	$2^{106.66}$	$2^{107.13}$	$2^{97.94}$	$2^{99.13}$
128	32	$2^{135.91}$	$2^{136.38}$	$2^{126.36}$	$2^{127.54}$
128	40	$2^{164.87}$	$2^{165.34}$	$2^{154.67}$	$2^{155.87}$
256	16	$2^{85.22}$	$2^{85.69}$	$2^{77.63}$	$2^{78.82}$
256	24	$2^{118.97}$	$2^{119.44}$	$2^{110.21}$	$2^{111.41}$
256	32	$2^{152.21}$	$2^{152.69}$	$2^{142.63}$	$2^{143.83}$

Table 2.3: Cost of Exhaustive search on the $\mathcal{MQ}$ problem using Grover's algorithm

The HybridF5 oracle.

The Hybrid Approach includes partial evaluation of $n - k$ variables. The cost of this part can be computed similarly as for the $\mathcal{MQ}$ oracle. We found that for this part we need

$6(n-k)km$ multiplications and the same amount of additions. The depth of this part of the circuit is $6(n-k)$ times the depth of a multiplication and $6(n-k)$ times the depth of an addition.

The rest of the cost of the Hybrid Approach comes from implementing Strassen's algorithm on the Macaulay matrix of size $\binom{k+D_{reg}-1}{D_{reg}}$ in a quantum circuit. The cost is approximately $5\binom{k+D_{reg}-1}{D_{reg}}^{\log_2 7}$ field additions which can be realized in circuit depth $3\log_2\binom{k+D_{reg}-1}{D_{reg}}$ times the depth of an addition.

Using the formulas:

$$\begin{aligned} Gates(HybF5Grover) &= \left\lfloor \frac{\pi}{4} 2^{\lfloor \log_2 q \rfloor \frac{n-k}{2}} \right\rfloor \cdot (Gates(U_{HybF5}) + Gates(U_s)), \\ Depth(HybF5Grover) &= \left\lfloor \frac{\pi}{4} 2^{\lfloor \log_2 q \rfloor \frac{n-k}{2}} \right\rfloor \cdot (Depth(U_{HybF5}) + Depth(U_s)), \end{aligned} \tag{2.9}$$

and the results from Subsections 2.3.1 and 2.3.2 we obtain an estimate of the cost of the Hybrid F5 algorithm with Grover's search algorithm. The results are summarized in Table 2.4.

The Crossbread oracle.

Similarly as the HybridF5 oracle, the Crossbread oracle includes partial evaluation of $n-k$ variables, so this cost is the same. For each enumeration, the Strassen's algorithm is applied on a matrix of size $\binom{k+d-1}{d}$ for a small d which costs $5\binom{k+d-1}{d}^{\log_2 7}$ field additions and total depth $3\log_2\binom{k+d-1}{d}$ times the depth of an addition. An important feature of the algorithm is that it can be split into two distinct parts: Sparse linear algebra on the Macaulay matrix, and enumeration plus dense linear algebra to check the consistency of the smaller system obtained from the kernel elements of $M^{(k)}_{D,d}$. The first part, that is more memory demanding can always be performed on a classical computer, and the second part which can make use of Grover's algorithm can be performed on a quantum computer. This is undoubtedly a big advantage over the quantum version of the Hybrid Approach, albeit the later is in theory faster.

The cost of the entire algorithm for various parameters is given in Table 2.5.

The BooleanSolve oracle.

For the BooleanSolve oracle, the cost of the partial evaluation of $n-k$ is the same as for the previous oracles. The rest comes from consistency checking of a system of size $\binom{k}{D_{reg}}$ using a sparse linear algebra technique from [34]. For simplicity, we will lower bound the cost of this part by $\binom{k}{D_{reg}}^2 \log^2\binom{k}{D_{reg}}$ additions (xor) in $\mathbb{F}_2$ of linear depth $\binom{k}{D_{reg}}$.

The cost of the entire algorithm for various parameters is given in Table 2.6.

			Gates		Depth	
q	n	k	T	Clifford	T	Total
2	120	59	$2^{54.28}$	$2^{132.95}$	$2^{42.50}$	$2^{43.65}$
2	128	63	$2^{56.56}$	$2^{136.83}$	$2^{44.59}$	$2^{45.74}$
2	168	83	$2^{67.73}$	$2^{166.86}$	$2^{54.99}$	$2^{56.14}$
2	192	95	$2^{74.31}$	$2^{189.50}$	$2^{61.18}$	$2^{62.34}$
2	232	115	$2^{85.12}$	$2^{219.10}$	$2^{71.46}$	$2^{72.62}$
4	72	35	$2^{60.16}$	$2^{127.84}$	$2^{49.27}$	$2^{50.39}$
4	96	47	$2^{73.40}$	$2^{158.07}$	$2^{61.69}$	$2^{62.80}$
4	104	51	$2^{77.74}$	$2^{164.57}$	$2^{65.80}$	$2^{66.92}$
4	128	63	$2^{90.64}$	$2^{204.70}$	$2^{78.10}$	$2^{79.22}$
4	136	67	$2^{94.90}$	$2^{211.07}$	$2^{82.19}$	$2^{83.32}$
16	48	23	$2^{73.00}$	$2^{113.80}$	$2^{62.70}$	$2^{63.77}$
16	64	31	$2^{90.24}$	$2^{145.32}$	$2^{79.11}$	$2^{80.20}$
16	72	35	$2^{98.74}$	$2^{165.84}$	$2^{87.28}$	$2^{88.36}$
16	88	43	$2^{115.61}$	$2^{197.35}$	$2^{103.57}$	$2^{104.66}$
31	40	18	$2^{113.40}$	$2^{113.77}$	$2^{72.64}$	$2^{73.83}$
31	48	20	$2^{131.12}$	$2^{131.49}$	$2^{87.79}$	$2^{88.98}$
31	56	22	$2^{148.41}$	$2^{148.78}$	$2^{102.89}$	$2^{104.08}$
31	64	28	$2^{166.05}$	$2^{166.41}$	$2^{108.04}$	$2^{109.23}$
31	72	30	$2^{183.23}$	$2^{183.59}$	$2^{123.09}$	$2^{124.28}$
32	40	19	$2^{75.22}$	$2^{113.09}$	$2^{65.26}$	$2^{66.33}$
32	56	27	$2^{96.67}$	$2^{149.02}$	$2^{85.74}$	$2^{86.82}$
32	64	31	$2^{107.25}$	$2^{162.14}$	$2^{95.94}$	$2^{97.01}$
32	72	35	$2^{117.75}$	$2^{184.66}$	$2^{106.10}$	$2^{107.18}$
64	40	19	$2^{86.14}$	$2^{123.85}$	$2^{76.02}$	$2^{77.09}$
64	48	23	$2^{98.92}$	$2^{139.38}$	$2^{88.29}$	$2^{89.36}$
64	56	27	$2^{111.59}$	$2^{163.78}$	$2^{100.51}$	$2^{101.58}$
64	64	31	$2^{124.16}$	$2^{178.90}$	$2^{112.70}$	$2^{113.77}$
128	32	15	$2^{82.03}$	$2^{108.29}$	$2^{72.43}$	$2^{73.49}$
128	40	19	$2^{96.99}$	$2^{134.57}$	$2^{86.75}$	$2^{87.82}$
128	48	23	$2^{111.78}$	$2^{152.11}$	$2^{101.01}$	$2^{102.08}$
128	56	27	$2^{126.44}$	$2^{178.51}$	$2^{115.23}$	$2^{116.30}$
256	32	15	$2^{90.84}$	$2^{116.99}$	$2^{81.12}$	$2^{82.19}$
256	40	19	$2^{107.80}$	$2^{145.27}$	$2^{97.44}$	$2^{98.51}$
256	48	23	$2^{124.58}$	$2^{164.80}$	$2^{113.70}$	$2^{114.77}$
256	56	27	$2^{141.25}$	$2^{193.20}$	$2^{129.93}$	$2^{130.99}$

Table 2.4: Cost of applying the Hybrid F5 algorithm on the $\mathcal{MQ}$ problem using Grover's algorithm

			Classical part	Gates		Depth	
q	n	k	Field. op.	T	Clifford	T	Total
2	128	21	$2^{69.79}$	$2^{76.69}$	$2^{77.05}$	$2^{66.20}$	$2^{67.31}$
2	160	25	$2^{91.87}$	$2^{91.60}$	$2^{91.96}$	$2^{80.54}$	$2^{81.64}$
2	192	27	$2^{105.09}$	$2^{107.26}$	$2^{107.62}$	$2^{95.82}$	$2^{96.92}$
2	224	30	$2^{118.36}$	$2^{122.36}$	$2^{122.73}$	$2^{110.55}$	$2^{111.65}$
2	296	35	$2^{154.47}$	$2^{156.92}$	$2^{157.28}$	$2^{144.46}$	$2^{145.57}$
4	88	19	$2^{92.65}$	$2^{92.46}$	$2^{92.88}$	$2^{82.45}$	$2^{83.59}$
4	96	19	$2^{88.24}$	$2^{100.74}$	$2^{101.16}$	$2^{90.59}$	$2^{91.74}$
4	120	23	$2^{115.69}$	$2^{121.67}$	$2^{122.09}$	$2^{110.93}$	$2^{112.07}$
4	128	24	$2^{124.76}$	$2^{128.93}$	$2^{129.34}$	$2^{118.02}$	$2^{119.17}$
4	160	28	$2^{154.32}$	$2^{157.81}$	$2^{158.23}$	$2^{146.36}$	$2^{147.50}$
16	48	17	$2^{85.02}$	$2^{84.87}$	$2^{85.33}$	$2^{75.77}$	$2^{76.93}$
16	56	19	$2^{94.79}$	$2^{97.51}$	$2^{97.96}$	$2^{88.01}$	$2^{89.18}$
16	72	23	$2^{123.38}$	$2^{122.54}$	$2^{123.00}$	$2^{112.41}$	$2^{113.57}$
16	80	25	$2^{133.03}$	$2^{134.98}$	$2^{135.43}$	$2^{124.57}$	$2^{125.74}$
16	96	28	$2^{156.92}$	$2^{161.71}$	$2^{162.16}$	$2^{150.86}$	$2^{152.03}$
31	48	20	$2^{97.57}$	$2^{99.27}$	$2^{99.63}$	$2^{89.33}$	$2^{90.52}$
31	56	22	$2^{112.01}$	$2^{114.76}$	$2^{115.13}$	$2^{104.47}$	$2^{105.66}$
31	64	24	$2^{126.43}$	$2^{130.17}$	$2^{130.54}$	$2^{119.57}$	$2^{120.76}$
31	72	27	$2^{140.84}$	$2^{143.07}$	$2^{143.43}$	$2^{132.12}$	$2^{133.31}$
31	80	28	$2^{151.00}$	$2^{160.81}$	$2^{161.18}$	$2^{149.67}$	$2^{150.86}$
32	48	19	$2^{93.55}$	$2^{95.95}$	$2^{96.41}$	$2^{86.65}$	$2^{87.83}$
32	64	24	$2^{126.43}$	$2^{124.65}$	$2^{125.12}$	$2^{114.60}$	$2^{115.78}$
32	72	26	$2^{136.64}$	$2^{140.14}$	$2^{140.60}$	$2^{129.79}$	$2^{130.97}$
32	80	28	$2^{151.00}$	$2^{155.57}$	$2^{156.03}$	$2^{144.96}$	$2^{146.14}$
32	88	29	$2^{156.56}$	$2^{173.44}$	$2^{173.90}$	$2^{162.63}$	$2^{163.81}$
64	40	18	$2^{86.85}$	$2^{89.13}$	$2^{89.60}$	$2^{80.17}$	$2^{81.34}$
64	48	21	$2^{105.20}$	$2^{104.90}$	$2^{105.37}$	$2^{95.45}$	$2^{96.63}$
64	56	24	$2^{119.83}$	$2^{120.56}$	$2^{121.03}$	$2^{110.69}$	$2^{111.87}$
64	64	27	$2^{138.24}$	$2^{136.13}$	$2^{136.60}$	$2^{125.89}$	$2^{127.07}$
64	72	27	$2^{140.84}$	$2^{160.58}$	$2^{161.05}$	$2^{150.14}$	$2^{151.32}$
128	40	19	$2^{90.48}$	$2^{96.99}$	$2^{97.47}$	$2^{87.93}$	$2^{89.11}$
128	56	26	$2^{130.80}$	$2^{129.93}$	$2^{130.41}$	$2^{119.94}$	$2^{121.12}$
128	64	27	$2^{138.24}$	$2^{154.98}$	$2^{155.45}$	$2^{144.71}$	$2^{145.89}$
128	72	27	$2^{140.84}$	$2^{183.43}$	$2^{183.90}$	$2^{172.96}$	$2^{174.15}$
256	32	15	$2^{68.54}$	$2^{90.84}$	$2^{91.32}$	$2^{82.39}$	$2^{83.58}$
256	48	23	$2^{112.38}$	$2^{124.58}$	$2^{125.06}$	$2^{114.96}$	$2^{116.14}$
256	64	27	$2^{138.24}$	$2^{173.79}$	$2^{174.26}$	$2^{163.48}$	$2^{164.67}$

Table 2.5: Cost of applying the Crossbread algorithm on the $\mathcal{MQ}$ problem using Grover's algorithm

		Gates		Depth	
n	k	T	Clifford	T	Total
144	71	$2^{61.06}$	$2^{112.88}$	$2^{48.76}$	$2^{69.46}$
160	79	$2^{65.52}$	$2^{119.58}$	$2^{52.92}$	$2^{74.75}$
192	95	$2^{74.31}$	$2^{138.99}$	$2^{61.18}$	$2^{88.25}$
208	103	$2^{78.66}$	$2^{151.94}$	$2^{65.30}$	$2^{96.57}$
256	127	$2^{91.55}$	$2^{177.46}$	$2^{77.60}$	$2^{115.13}$
264	131	$2^{93.68}$	$2^{180.51}$	$2^{79.65}$	$2^{117.65}$

Table 2.6: Cost of BooleanSolve on the $\mathcal{MQ}$ problem using Grover's algorithm

3

The Sakumoto-Shirai-Hiwatari (SSH) 5-pass IDS scheme

3.1 Description of the SSH 5-pass IDS

In [43], Sakumoto, Shirai, and Hiwatari proposed two new identification schemes, a 3-pass and a 5-pass IDS, based on the intractability of the $\mathcal{MQ}$ problem. In these specifications, and documents related to this submission, we will refer to identification schemes from [43] as the SSH 3-pass and 5-pass schemes.

Unlike previous public key schemes, the SSH schemes provably rely only on the $\mathcal{MQ}$ problem (and the security of the commitment scheme), and not on other related problems in multivariate cryptography such as the Isomorphism of Polynomials (IP) [51], the related Extended IP [25] and IP with partial knowledge [55] problems or the MinRank problem [22, 29].

The main idea from [43] is a clever splitting of the secret, that relies on the polar form of the function $\mathbf{F}$. With this technique, the secret $\mathbf{s}$ is split into $\mathbf{s} = \mathbf{r}_0 + \mathbf{r}_1$, and the public $\mathbf{v} = \mathbf{F}(\mathbf{s})$ can be represented as $\mathbf{v} = \mathbf{F}(\mathbf{r}_0) + \mathbf{F}(\mathbf{r}_1) + \mathbf{G}(\mathbf{r}_0, \mathbf{r}_1)$. In order for the polar form not to depend on both shares of the secret, $\mathbf{r}_0$ and $\mathbf{F}(\mathbf{r}_0)$ are further split as $\alpha\mathbf{r}_0 = \mathbf{t}_0 + \mathbf{t}_1$ and $\alpha\mathbf{F}(\mathbf{r}_0) = \mathbf{e}_0 + \mathbf{e}_1$. Now, because of the bilinearity of the polar form it holds that $\alpha\mathbf{v} = (\mathbf{e}_1 + \alpha\mathbf{F}(\mathbf{r}_1) + \mathbf{G}(\mathbf{t}_1, \mathbf{r}_1)) + (\mathbf{e}_0 + \mathbf{G}(\mathbf{t}_0, \mathbf{r}_1))$, and from only one of the two summands, represented by $(\mathbf{r}_1, \mathbf{t}_1, \mathbf{e}_1)$ and $(\mathbf{r}_1, \mathbf{t}_0, \mathbf{e}_0)$, nothing can be learned about the secret $\mathbf{s}$.

Let $(\mathsf{pk}, \mathsf{sk}) = ((\mathbf{F}, \mathbf{v}), \mathbf{s}) \in R_{\mathcal{MQ}}$ be the public and private keys of the prover $\mathcal{P}$ (i.e., key generation just samples from the $\mathcal{MQ}$ relation). The SSH 5-pass IDS from [43] is given in Figure 3.1.

3.2 Properties of the SSH 5-pass IDS

The following theorem summarizes the properties of the SSH 5-pass IDS.

Theorem 3.1. *The 5-pass identification scheme of Sakumoto, Shirai, and Hiwatari [43]:*

1. *Has key relation $R_{\mathcal{MQ}(m,n,q)}$,*
2. *Is KOW if the $\mathcal{MQ}$ search problem is hard on average,*
3. *Is perfectly correct,*
4. *Is computationally HVZK when the commitment scheme Com is computationally hiding,*
5. *Is argument of knowledge for $R_{\mathcal{MQ}(m,n,q)}$ with knowledge error $\frac{1}{2} + \frac{1}{2q}$ when the commitment scheme Com is computationally binding,*

$\mathcal{P}(\mathsf{pk}, \mathsf{sk})$		$\mathcal{V}(\mathsf{pk})$
//setup		
$\mathbf{r}_0, \mathbf{t}_0 \leftarrow_R \mathbb{F}_q^n, \mathbf{e}_0 \leftarrow_R \mathbb{F}_q^m$		
$\mathbf{r}_1 \leftarrow \mathbf{s} - \mathbf{r}_0$		
//commit		
$c_0 \leftarrow Com(\mathbf{r}_0, \mathbf{t}_0, \mathbf{e}_0)$		
$c_1 \leftarrow Com(\mathbf{r}_1, \mathbf{G}(\mathbf{t}_0, \mathbf{r}_1) + \mathbf{e}_0)$	$\xrightarrow{\mathsf{com} = (c_0, c_1)}$	*//challenge* 1 $\alpha \leftarrow_R \mathbb{F}_q$
//first response	$\xleftarrow{\mathsf{ch}_1 = \alpha}$	
$\mathbf{t}_1 \leftarrow \alpha\mathbf{r}_0 - \mathbf{t}_0$		
$\mathbf{e}_1 \leftarrow \alpha\mathbf{F}(\mathbf{r}_0) - \mathbf{e}_0$	$\xrightarrow{\mathsf{resp}_1 = (\mathbf{t}_1, \mathbf{e}_1)}$	*//challenge* 2 $\mathsf{ch}_2 \leftarrow_R \{0, 1\}$
//second response	$\xleftarrow{\mathsf{ch}_2}$	
If $\mathsf{ch}_2 = 0,\ \mathsf{resp}_2 \leftarrow \mathbf{r}_0$		
Else $\mathsf{resp}_2 \leftarrow \mathbf{r}_1$	$\xrightarrow{\mathsf{resp}_2}$	*//verify* If $\mathsf{ch}_2 = 0$, parse $\mathsf{resp}_2 = \mathbf{r}_0$, check $c_0 \stackrel{?}{=} Com(\mathbf{r}_0, \alpha\mathbf{r}_0 - \mathbf{t}_1, \alpha\mathbf{F}(\mathbf{r}_0) - \mathbf{e}_1)$ Else, parse $\mathsf{resp}_2 = \mathbf{r}_1$, check $c_1 \stackrel{?}{=} Com(\mathbf{r}_1, \alpha(\mathbf{v} - \mathbf{F}(\mathbf{r}_1)) - \mathbf{G}(\mathbf{t}_1, \mathbf{r}_1) - \mathbf{e}_1)$

Fig. 3.1: The SSH 5-pass IDS by Sakumoto, Shirai, and Hiwatari [43]

6. *Is sound with soundness error* $\frac{1}{2} + \frac{1}{2q}$ *when the commitment scheme Com is computationally binding,*
7. *Has a q2-Extractor when the commitment scheme Com is computationally binding.*

The first statement holds by construction. The second statement follows directly from the first. The third, a stronger version of the fourth[1] and the fifth were proven in [43]. The last two statements were proven in [16].

[1] Sakumoto et al. [43] proved that their 5-pass scheme is statistically zero knowledge when the commitment scheme *Com* is statistically hiding which implies (honest-verifier) zero knowledge. Relaxing the requirements of *Com* to computationally hiding, weakens the result to computationally HVZK, since now, it is possible to distinguish (albeit only with negligible probability) whether the commitment was produced in a valid run of the protocol.

4

The Fiat-Shamir Transform

The Fiat-Shamir paradigm [30] for transforming canonical 3-pass identification schemes to signatures has been one the most popular methods for obtaining classically secure signature schemes. In this chapter, we present the transform, known results about its security, as well as its limitations.

4.1 Description of the Fiat-Shamir Transform

In what follows, let $\mathsf{IDS}^r = (\mathsf{KGen}_{\mathsf{IDS}}, \mathcal{P}^r, \mathcal{V}^r)$ denote the parallel composition of r rounds of the identification scheme $\mathsf{IDS} = (\mathsf{KGen}_{\mathsf{IDS}}, \mathcal{P}, \mathcal{V})$.

Construction 4.1 (Fiat-Shamir transform [30]) *Let $k \in \mathbb{N}$ the security parameter, $\mathsf{IDS} = (\mathsf{KGen}_{\mathsf{IDS}}, \mathcal{P}, \mathcal{V})$, where $\mathcal{P} = (\mathcal{P}_0, \mathcal{P}_1)$, $\mathcal{V} = (\mathsf{ChS}, \mathsf{Vf}_{\mathsf{IDS}})$ a canonical 3-pass Identification scheme that achieves soundness with soundness error κ. Select r, the number of (parallel) rounds of IDS, such that $\kappa^r = \mathrm{negl}(k)$, and that the challenge space C^r of the composition IDS^r, has exponential size in k. Moreover, select a cryptographic hash function $H : \{0,1\}^* \to \mathsf{C}^r$.*

A signature scheme derived from IDS via the Fiat-Shamir transform is a triplet of algorithms $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ defined as in Figures 4.1,4.2 and 4.3.

```
$\mathsf{KGen}()$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\mathsf{IDS}}$
**Return** $(\mathsf{pk}, \mathsf{sk})$
```

Fig. 4.1: Fiat-Shamir key generation

```
$\mathsf{Sign}(\mathsf{sk}, M)$
$\textbf{For } j \in \{1, \ldots, r\} \textbf{ do}$
    $(\mathsf{state}^{(j)}, \mathsf{com}^{(j)}) \leftarrow \mathcal{P}_0(\mathsf{sk})$
$\mathsf{state} := (\mathsf{state}^{(1)}, \ldots, \mathsf{state}^{(r)})$
$\mathsf{com} := (\mathsf{com}^{(1)}, \ldots, \mathsf{com}^{(r)})$
$\sigma_0 := \mathsf{com}$
$\mathsf{ch} \leftarrow H(\mathsf{pk}, M, \sigma_0)$
Parse $\mathsf{ch}$ as $\mathsf{ch} = (\mathsf{ch}^{(1)}, \mathsf{ch}^{(2)}, \ldots, \mathsf{ch}^{(r)}),\ \mathsf{ch}^{(j)} \in \mathsf{C}$
$\textbf{For } j \in \{1, \ldots, r\} \textbf{ do}$
    $\mathsf{resp}^{(j)} \leftarrow \mathcal{P}_1(\mathsf{state}^{(j)}, \mathsf{ch}^{(j)})$
$\mathsf{resp} := (\mathsf{resp}^{(1)}, \ldots, \mathsf{resp}^{(r)})$
$\sigma_1 := \mathsf{resp}$
$\textbf{Return } \sigma = (\sigma_0, \sigma_1)$
```

Fig. 4.2: Fiat-Shamir signature generation

```
$\mathsf{Vf}(\mathsf{pk}, \sigma, M)$
Parse $\sigma = (\sigma_0, \sigma_1)$
Parse $\sigma_0$ as $\sigma_0 = (\mathsf{com}^{(1)}, \ldots, \mathsf{com}^{(r)})$
$\mathsf{ch} \leftarrow H(\mathsf{pk}, M, \sigma_0)$
Parse $\mathsf{ch}$ as $\mathsf{ch} = (\mathsf{ch}^{(1)}, \mathsf{ch}^{(2)}, \ldots, \mathsf{ch}^{(r)}),\ \mathsf{ch}^{(j)} \in \mathsf{C}$
Parse $\sigma_1$ as $\sigma_1 = (\mathsf{resp}^{(1)}, \ldots, \mathsf{resp}^{(r)})$
$\textbf{For } j \in \{1, \ldots, r\} \textbf{ do}$
    $b^{(j)} \leftarrow \mathsf{Vf}_{\mathsf{IDS}}(\mathsf{pk}, \mathsf{com}^{(j)}, \mathsf{ch}^{(j)}, \mathsf{resp}^{(j)})$
$b \leftarrow b^{(1)} \wedge b^{(2)} \wedge \cdots \wedge b^{(r)}$
$\textbf{Return } b$
```

Fig. 4.3: Fiat-Shamir signature verification

4.2 Security of the Fiat-Shamir Transform

The security of the Fiat-Shamir transform has been investigated for over two decades. The first security proof of the transform was given in the seminal paper of Pointcheval and Stern [52]. They showed that assuming honest-verifier zero knowledge and special soundness of the identification scheme, Construction 5.1 gives EU-CMA secure signatures. Their proof is in the random oracle model and is based on the (now famous) forking lemma. The two main techniques introduced in the forking lemma are rewinding of the adversary and adaptively programming the random oracle. While these have proven to be quite powerful techniques in ROM reductions, they come with a drawback - the reduction is not tight - there is loss of factor the number of adversary's random oracle queries.

Later, Ohta and Okamoto [50] provide a different proof using a modular technique and similar assumptions on the identification scheme. Abdalla *et al.*[1] show that a signature via the Fiat-Shamir transform is EU-CMA if and only if the identification scheme is IMP-PA, thus minimizing the needed assumption on the identification scheme.

Lately, the main two techniques from the forking lemma cause even more problems in the quantum-accessible random oracle model (QROM) and showing the Fiat-Shamir transform secure in the QROM has proven to be a tedious task (see [2, 23, 58]). Until around 3 weeks before the deadline for Second round tweaks, there were only two known ways to show security of the Fiat-Shamir transform in the QROM setting - using oblivious commitments [23], and using so called lossy identification schemes [40]. In the first approach, the need for rewinding is replaced by introducing an additional trapdoor assumption on the commitments, which in itself is a very strong and problematic assumption. The second approach seems more realistic, since it only requires existence of 'fake' public keys indistinguishable from the 'real' ones, for which on the other hand it is hard to find a matching private key. Just a few days ago, a paper claiming to have proven the security of Fiat-Shamir appeared on eprint [26]. The claims are certainly very exciting, but due to the short time until the submission deadline, we will not further comment these results. Note that they potentially open the doors for proving the security of MQDSS in the QROM as well, provided it is possible to extend their results to 5-pass schemes.

Very recently, multi-user security of the Fiat-Shamir transform has been investigated in [41]. Although the authors of [41] attempt to provide a more general framework for multi-user security (previously, tight results have been obtained only for Schnorr like signatures [10]), the assumptions on the IDS still seem too strong to be applicable on many post-quantum schemes. Thus for now, the only general result remains the one from [32], with a loss of factor - the number of users in the scenario.

5

The Fiat-Shamir Transform for 5-pass Identification Schemes

For several intractability assumptions, the most efficient IDS are five pass, i.e. IDS where a transcript consists of five messages. Here, efficiency refers to the size of all communication of sufficient rounds to make the soundness error negligible. This becomes especially relevant when one wants to turn an IDS into a signature scheme as it is closely related to the signature size of the resulting scheme.

As said in the Preliminaries (Chapter 1), the most common 5-pass identification schemes in the literature are those with challenge spaces C_1 and C_2 restricted to q and 2 respectively, that we called $q2$-Identification Schemes. In this chapter, we restrict our attention to such schemes and describe a transformation from passively secure $q2\text{-}IDS$ to unforgeable signatures. The transformation is a direct generalization of the Fiat-Shamir transform for 3-pass schemes (see Chapter 4). The description and security argument follow closely the one from [16], where we first introduced MQDSS.

Note that the Fiat-Shamir transform can be generalized to more general canonical $2n+1$ schemes with non-binary challenge spaces. However, this makes the description and proofs unnecessarily complex, especially because such schemes are not at all common in the literature.

5.1 A Fiat-Shamir transform for $q2$-Identification Schemes

As in the previous chapter, let $\mathsf{IDS}^r = (\mathsf{KGen}_{\mathsf{IDS}}, \mathcal{P}^r, \mathcal{V}^r)$ denote the parallel composition of r rounds of the identification scheme $\mathsf{IDS} = (\mathsf{KGen}_{\mathsf{IDS}}, \mathcal{P}, \mathcal{V})$.

Construction 5.1 (Fiat-Shamir transform for $q2$-Identification Schemes) *Let $k \in \mathbb{N}$ be the security parameter and let $\mathsf{IDS} = (\mathsf{KGen}_{\mathsf{IDS}}, \mathcal{P}, \mathcal{V})$, where $\mathcal{P} = (\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2)$, $\mathcal{V} = (\mathsf{ChS}_1, \mathsf{ChS}_2, \mathsf{Vf}_{\mathsf{IDS}})$ be a $q2$-Identification Scheme that achieves soundness with soundness error κ. Select r, the number of (parallel) rounds of IDS, such that $\kappa^r = \mathrm{negl}(k)$, and that the challenge spaces C_1^r and C_2^r of the composition IDS^r, have exponential size in k. Moreover, select two cryptographic hash functions $H_1 : \{0,1\}^* \to \mathsf{C}_1^r$, $H_2 : \{0,1\}^* \to \mathsf{C}_2^r$.*

A $q2$-signature scheme $q2\text{-}\mathsf{Dss}(1^k)$ is a triplet of algorithms $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ defined as in Figures 5.1,5.1 and 5.3.

```
$\mathsf{KGen}()$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\mathsf{IDS}}$
**Return** $(\mathsf{pk}, \mathsf{sk})$
```

Fig. 5.1: $q2$-signature scheme: Key generation

```
$\mathsf{Sign}(\mathsf{sk}, M)$
**For** $j \in \{1, \ldots, r\}$ **do**
    $(\mathsf{state}^{(j)}, \mathsf{com}^{(j)}) \leftarrow \mathcal{P}_0(\mathsf{sk})$
$\mathsf{state} := (\mathsf{state}^{(1)}, \ldots, \mathsf{state}^{(r)})$
$\mathsf{com} := (\mathsf{com}^{(1)}, \ldots, \mathsf{com}^{(r)})$
$\sigma_0 := \mathsf{com}$
$\mathsf{h}_1 \leftarrow H_1(\mathsf{pk}, M, \sigma_0)$
Parse $\mathsf{h}_1$ as $\mathsf{h}_1 = (\mathsf{ch}_1^{(1)}, \mathsf{ch}_1^{(2)}, \ldots, \mathsf{ch}_1^{(r)}),\ \mathsf{ch}_1^{(j)} \in \mathsf{C}_1$
**For** $j \in \{1, \ldots, r\}$ **do**
    $(\mathsf{state}^{(j)}, \mathsf{resp}_1^{(j)}) \leftarrow \mathcal{P}_1(\mathsf{state}^{(j)}, \mathsf{ch}_1^{(j)})$
$\mathsf{state} := (\mathsf{state}^{(1)}, \ldots, \mathsf{state}^{(r)})$
$\mathsf{resp}_1 := (\mathsf{resp}_1^{(1)}, \ldots, \mathsf{resp}_1^{(r)})$
$\sigma_1 := \mathsf{resp}_1$
$\mathsf{h}_2 \leftarrow H_2(\mathsf{pk}, M, \sigma_0, \sigma_1)$
Parse $\mathsf{h}_2$ as $\mathsf{h}_2 = (\mathsf{ch}_2^{(1)}, \mathsf{ch}_2^{(2)}, \ldots, \mathsf{ch}_2^{(r)}),\ \mathsf{ch}_2^{(j)} \in \mathsf{C}_2$
**For** $j \in \{1, \ldots, r\}$ **do**
    $\mathsf{resp}_2^{(j)} \leftarrow \mathcal{P}_2(\mathsf{state}^{(j)}, \mathsf{ch}_2^{(j)})$
$\mathsf{resp}_2 := (\mathsf{resp}_2^{(1)}, \ldots, \mathsf{resp}_2^{(r)})$
$\sigma_2 := \mathsf{resp}_2$
**Return** $\sigma = (\sigma_0, \sigma_1, \sigma_2)$
```

Fig. 5.2: $q2$-signature scheme: Signature generation

The correctness of the scheme follows immediately from the correctness of the IDS.

5.2 Security of $q2$-signature schemes.

The security of the above transform was proven in [16]. The proof is in the random oracle model and builds on techniques introduced by Pointcheval and Stern [52]. Namely, we generalized the well known Forking Lemma and showed that a particular type of rewinding of the adversary together with adaptive programming of the random oracles is useful for showing EU-CMA security. For completeness of this document, we include the complete security reduction in Appendix A[1].

In summary, the following theorem holds for $q2$-signature schemes:

[1] In [16], the claim was proven for a stronger assumption for the $q2$-IDS- being Honest Verifier Zero-Knowledge. However this stronger assumption is not actually needed, and '5.2 holds even for computational HVZK.

```
$\mathsf{Vf}(\mathsf{pk}, \sigma, M)$

Parse $\sigma = (\sigma_0, \sigma_1, \sigma_2)$
Parse $\sigma_0$ as $\sigma_0 = (\mathsf{com}^{(1)}, \ldots, \mathsf{com}^{(r)})$
$\mathsf{h}_1 \leftarrow H_1(\mathsf{pk}, M, \sigma_0)$
Parse $\mathsf{h}_1$ as $\mathsf{h}_1 = (\mathsf{ch}_1^{(1)}, \mathsf{ch}_1^{(2)}, \ldots, \mathsf{ch}_1^{(r)}),\ \mathsf{ch}_1^{(j)} \in \mathsf{C}_1$
Parse $\sigma_1$ as $\sigma_1 = (\mathsf{resp}_1^{(1)}, \ldots, \mathsf{resp}_1^{(r)})$
$\mathsf{h}_2 \leftarrow H_2(\mathsf{pk}, M, \sigma_0, \sigma_1)$
Parse $\mathsf{h}_2$ as $\mathsf{h}_2 = (\mathsf{ch}_2^{(1)}, \mathsf{ch}_2^{(2)}, \ldots, \mathsf{ch}_2^{(r)}),\ \mathsf{ch}_2^{(j)} \in \mathsf{C}_2$
Parse $\sigma_2$ as $\sigma_2 = (\mathsf{resp}_2^{(1)}, \ldots, \mathsf{resp}_2^{(r)})$
For $j \in \{1, \ldots, r\}$ do
    $b^{(j)} \leftarrow \mathsf{Vf}_{\mathsf{IDS}}(\mathsf{pk}, \mathsf{com}^{(j)}, \mathsf{ch}_1^{(j)}, \mathsf{resp}_1^{(j)}, \mathsf{ch}_2^{(j)}, \mathsf{resp}_2^{(j)})$
$b \leftarrow b^{(1)} \wedge b^{(2)} \wedge \cdots \wedge b^{(r)}$
Return $b$
```

Fig. 5.3: $q2$-signature scheme: Signature verification

Theorem 5.2 (EU-CMA security of $q2$-signature schemes [16]). *Let $k \in \mathbb{N}$, $\mathsf{IDS}(1^k)$ a $q2$-IDS that has a key relation R, is KOW secure, is computationally honest-verifier zero-knowledge, and has a $q2$-extractor $\mathcal{E}$. Then $q2\text{-}\mathsf{Dss}(1^k)$, the $q2$-signature scheme derived applying Construction 5.1 is existentially unforgeable under adaptive chosen message attacks.*

Part II

MQDSS Specifications

6

Notations

Throughout this part we will use the standard mathematical notations introduced in Section 1.1. In addition, in Chapter 9 we will use the following notations:

- $a + b$ - sum of a and b
- $a - b$ - difference of a and b
- $a \cdot b$ - product of a and b
- a/b - quotient of a and b
- $\log_2 a$ - logarithm to the base 2 of a
- $a \mod b$ - the non-negative remainder of the integer division of a and b
- $a|b$ - a is a divisor of b
- $\lceil a \rceil$ - ceiling function, returns the smallest integer greater than or equal to a
- $\lfloor a \rfloor$ - floor function, returns the greatest integer larger than or equal to a

- $a \leftarrow b$ - assignment operator, a takes the value of b
- $a \ll b$ - logical left shift, with b being non-negative integer. It is equivalent to $a \cdot 2^b$
- $a \wedge b$ - logical and operator
- $a == b$ - logical equal test, returns true (1) if $a = b$ and false (0) if $a \neq b$
- $a <> b$ - logical non-equal test, returns true (1) if $a \neq b$ and false (0) if $a = b$

- $[\,]$ - empty array of bytes or bits
- $a[i]$ - the i-th element of the array a. The indexing of elements starts from 0.
- $[f(j)|j = 0..n-1]$ - array with elements $f(j)$, when j is iterated from 0 through $n-1$
- len(a) - returns the length of a (the number of elements in a if we consider it as an array)
- $a||b$ - concatenation of a and b. If we look at $a = [a[0], a[1], \ldots, a[l_a]]$ and $b = [b[0], b[1], \ldots, b[l_2]]$ as arrays, then $a||b$ is the array $[a[0], a[1], \ldots, a[l_a], b[0], b[1], \ldots, b[l_2]]$
- append(a, b) - appends the element b to the end of the array a
- subarray(a, b, c) - returns the subarray of a from index b to $c - 1$, i.e. returns $[a[b], \ldots, a[c-1]]$
- trunc(a, b) - truncates the b least significant bits of a, with a being a bit-array and b a non-negative integer. If we look at a in its array representation $a = [a[0], a[1], \ldots, a[l_a]]$, then trunc($a, b$)=$[a[0], a[1], \ldots, a[b-1]]$. It is equivalent to subarray($a, 0, b$), for a bit-array a.
- trim(a, b) - truncates the b most significant bits of a, with a being a bit-array and b a non-negative integer. It is equivalent to subarray(a,len(a)$-b$,len(a)), for a bit-array a.
- mask(a, b, c) - sets the bits $a[b], \ldots, a[c]$ to 0

- SHAKE256(*seed*, *outlen*) - interface for *outlen* bytes of SHAKE256 output on input *seed* as standardized in FIPS 202, the SHA-3 standard [48].
- SHAKE256absorb(*seed*) - interface for the absorb phase of SHAKE256 for extendable output
- SHAKE256squeeze(*state*)- interface for the squeeze phase of SHAKE256 for extendable output. To obtain extendable output, repeatedly call SHAKE256squeeze(*state*) until needed

7 MQDSS High Level Description

In this chapter, we define the signature scheme MQDSS-q-n in generic terms by describing the required parameters, the functions KGen, Sign and Vf, and the necessary auxiliary functions. In Chapter 8 we will provide concrete parameters and in Chapter 9 we provide a detailed instantiations of the auxiliary functions. A detailed low-level description will be given in Chapter 9.

MQDSS-q-n is a digital signature scheme consisting of three algorithms KGen, Sign and Vf, defined in Sections 7.2, 7.3 and 7.4. The global parameters and auxiliary functions are defined in Section 7.1.

7.1 MQDSS Parameters Description and Auxiliary Functions

Let k be the security parameter.
MQDSS-q-n uses the following additional parameters:

- A positive integer $n \in \mathbb{N}$ - the number of variables and equations of the system $\mathbf{F}$,
- A positive integer $q \in \mathbb{N}$ (a prime or a prime power) - the order of the finite field $\mathbb{F}_q$,
- A positive integer $r \in \mathbb{N}$ - the number of rounds. This parameter is normally $r = \left\lceil k/\log_2 \frac{2q}{q+1} \right\rceil$, but it can also be chosen independently.

MQDSS-q-n uses the following auxiliary functions:

- A pseudorandom generator $\mathrm{PRG}_{\mathsf{sk}} : \{0,1\}^k \to \{0,1\}^{4k}$ used to randomly generate four seeds.
- A pseudorandom generator $\mathrm{PRG}_{\mathbf{s}} : \{0,1\}^k \to \{0,1\}^{n\lceil \log_2 q \rceil}$ used to randomly generate the secret key.
- A pseudorandom generator $\mathrm{PRG}_{\rho} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^{4rk}$ used to generate pseudorandom values needed as input to the string commitment functions.
- A pseudorandom generator $\mathrm{PRG}_{\mathbf{rte}} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^{3rn\lceil \log_2 q \rceil}$ used to generate pseudorandom values during the signature generation.
- An extendable output function $\mathrm{XOF}_{\mathbf{F}} : \{0,1\}^k \to \{0,1\}^{F_{len}}$, where for $q = 2$, $F_{len} = n \cdot (\frac{n\cdot(n-1)}{2} + n)$ and for $q > 2$, $F_{len} = n \cdot (\frac{n\cdot(n+1)}{2} + n) \lceil \log_2 q \rceil$. This function is used for generating a multivariate system $\mathbf{F}$ by expanding a seed outputted by $\mathrm{PRG}_{\mathsf{sk}}$.
- Three cryptographic hash functions $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{2k}$, $H_1 : \{0,1\}^* \to \mathbb{F}_q{}^r$, and $H_2 : \{0,1\}^* \to \{0,1\}^r$.
- A string commitment function $Com_0 : \{0,1\}^{2k} \times \mathbb{F}_q{}^n \times \mathbb{F}_q{}^n \times \mathbb{F}_q{}^n \to \{0,1\}^{2k}$ and
- A string commitment function $Com_1 : \{0,1\}^{2k} \times \mathbb{F}_q{}^n \times \mathbb{F}_q{}^n \to \{0,1\}^{2k}$.

7.2 MQDSS Key Generation

The MQDSS-q-n key generation algorithm formally samples a $\mathcal{MQ}$ relation. Practically, the algorithm is realized as shown in Figure 7.1.

```
KGen()
$\mathsf{sk} \leftarrow_R \{0,1\}^k$
$S_{\mathbf{F}}, S_{\mathbf{s}}, S_\rho, S_{\mathbf{rte}} \leftarrow \mathrm{PRG}_{\mathsf{sk}}(\mathsf{sk})$
$\mathbf{F} \leftarrow \mathrm{XOF}_{\mathbf{F}}(S_{\mathbf{F}})$
$\mathbf{s} \leftarrow \mathrm{PRG}_{\mathbf{s}}(S_{\mathbf{s}})$
$\mathbf{v} \leftarrow \mathbf{F}(\mathbf{s})$
$\mathsf{pk} := (S_{\mathbf{F}}, \mathbf{v})$
Return $(\mathsf{pk}, \mathsf{sk})$
```

Fig. 7.1: MQDSS-q-n key generation

In more detail, given the security parameter k, the key generation algorithm $\mathsf{KGen}()$ performs the following operations:

- Randomly sample a secret key of k bits $\mathsf{sk} \leftarrow_R \{0,1\}^k$.
- Use the secret key sk as input (seed) to $\mathrm{PRG}_{\mathsf{sk}}$ to derive the following values:
  - $S_{\mathbf{F}}$, a seed of k bits from which the system parameter $\mathbf{F}$ is expanded;
  - $S_{\mathbf{s}}$, a seed of k bits from which the secret input to the $\mathcal{MQ}$ function is generated;
  - S_ρ, a seed of k bits that is used to generate random values $\rho_0^{(i)}$ and $\rho_1^{(i)}$, $i \in \{1,\ldots,r\}$ needed for the string commitment functions. Note that this seed is not yet needed during key generation, but is required during signing.
  - $S_{\mathbf{rte}}$, a seed of k bits that is used to sample all vectors $\mathbf{r}_0^{(i)}$, $\mathbf{t}_0^{(i)}$ and $\mathbf{e}_0^{(i)}$, $i \in \{1,\ldots,r\}$. Note that this seed is not yet needed during key generation, but is required during signing.
- Expand the seed $S_{\mathbf{F}}$ using $\mathrm{XOF}_{\mathbf{F}}$ to a F_{len} bits long string, where for $q = 2$, $F_{len} = n \cdot (\frac{n\cdot(n-1)}{2} + n)$ and for $q > 2$, $F_{len} = n \cdot (\frac{n\cdot(n+1)}{2} + n)\lceil \log_2 q \rceil$. Parse the pseudorandom string as an $\mathcal{MQ}$ system $\mathbf{F} \in \mathcal{MQ}(n, n, \mathbb{F}_q)$.
- Use the seed $S_{\mathbf{s}}$ as input to the $\mathrm{PRG}_{\mathbf{s}}$ to obtain $\mathbf{s}$, a string of length $n\lceil \log_2 q \rceil$ bits, that will be used as the secret input to the $\mathcal{MQ}$ function;
- Parse $\mathbf{s}$ as a vector $\mathbf{s} \in \mathbb{F}_q^n$, and evaluate the $\mathcal{MQ}$ system $\mathbf{F}(\mathbf{s})$ to obtain the vector $\mathbf{v} \in \mathbb{F}_q^n$.
- Set $\mathsf{pk} := (S_{\mathbf{F}}, \mathbf{v})$ as the public key.
- Return the public/secret key pair $(\mathsf{pk}, \mathsf{sk})$.

The obtained public key pk is of length $k + n\lceil \log_2 q \rceil$ bits, and the secret key sk of length k bits.

7.3 MQDSS Signature Generation

For the MQDSS-q-n signing procedure $\mathsf{Sign}()$, we assume as input a message $M \in \{0,1\}^*$ and a secret key sk. The signing procedure is given in Figure 7.2.

```
$\mathsf{Sign}(\mathsf{sk}, M)$
$S_{\mathbf{F}}, S_{\mathbf{s}}, S_{\rho}, S_{\mathbf{rte}} \leftarrow \mathrm{PRG}_{\mathsf{sk}}(\mathsf{sk})$
$\mathbf{F} \leftarrow \mathrm{XOF}_{\mathbf{F}}(S_{\mathbf{F}})$
$\mathbf{s} \leftarrow \mathrm{PRG}_{\mathbf{s}}(S_{\mathbf{s}})$
$\mathsf{pk} := (S_{\mathbf{F}}, \mathbf{F}(\mathbf{s}))$
$R \leftarrow \mathcal{H}(\mathsf{sk}||M)$
$D \leftarrow \mathcal{H}(\mathsf{pk}||R||M)$
$\rho_0^{(1)}, \ldots, \rho_0^{(r)}, \rho_1^{(1)}, \ldots, \rho_1^{(r)} \leftarrow \mathrm{PRG}_{\rho}(S_{\rho}, D)$
$\mathbf{r}_0^{(1)}, \ldots, \mathbf{r}_0^{(r)}, \mathbf{t}_0^{(1)}, \ldots, \mathbf{t}_0^{(r)}, \mathbf{e}_0^{(1)}, \ldots, \mathbf{e}_0^{(r)} \leftarrow \mathrm{PRG}_{\mathbf{rte}}(S_{\mathbf{rte}}, D)$
For $j \in \{1, \ldots, r\}$ do
    $\mathbf{r}_1^{(j)} \leftarrow \mathbf{s} - \mathbf{r}_0^{(j)}$
    $c_0^{(j)} \leftarrow Com_0(\rho_0^{(j)}, \mathbf{r}_0^{(j)}, \mathbf{t}_0^{(j)}, \mathbf{e}_0^{(j)})$
    $c_1^{(j)} \leftarrow Com_1(\rho_1^{(j)}, \mathbf{r}_1^{(j)}, \mathbf{G}(\mathbf{t}_0^{(j)}, \mathbf{r}_1^{(j)}) + \mathbf{e}_0^{(j)})$
    $\mathsf{com}^{(j)} := (c_0^{(j)}, c_1^{(j)})$
$\sigma_0 \leftarrow \mathcal{H}(\mathsf{com}^{(1)}||\mathsf{com}^{(2)}|| \ldots ||\mathsf{com}^{(r)})$
$\mathsf{ch}_1 \leftarrow H_1(D, \sigma_0)$
Parse $\mathsf{ch}_1$ as $\mathsf{ch}_1 = (\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(r)}), \alpha^{(j)} \in \mathbb{F}_q$
For $j \in \{1, \ldots, r\}$ do
    $\mathbf{t}_1^{(j)} \leftarrow \alpha^{(j)}\mathbf{r}_0^{(j)} - \mathbf{t}_0^{(j)}, \quad \mathbf{e}_1^{(j)} \leftarrow \alpha^{(j)}\mathbf{F}(\mathbf{r}_0^{(j)}) - \mathbf{e}_0^{(j)}$
    $\mathsf{resp}_1^{(j)} := (\mathbf{t}_1^{(j)}, \mathbf{e}_1^{(j)})$
$\sigma_1 \leftarrow (\mathsf{resp}_1^{(1)}||\mathsf{resp}_1^{(2)}|| \ldots ||\mathsf{resp}_1^{(r)})$
$\mathsf{ch}_2 \leftarrow H_2(D, \sigma_0, \mathsf{ch}_1, \sigma_1)$
Parse $\mathsf{ch}_2$ as $\mathsf{ch}_2 = (b^{(1)}, b^{(2)}, \ldots, b^{(r)}), b^{(j)} \in \{0, 1\}$
For $j \in \{1, \ldots, r\}$ do
    $\mathsf{resp}_2^{(j)} \leftarrow \mathbf{r}_{b^{(j)}}^{(j)}$
$\sigma_2 \leftarrow (\mathsf{resp}_2^{(1)}||\mathsf{resp}_2^{(2)}|| \ldots ||\mathsf{resp}_2^{(r)}||c_{1-b^{(1)}}^{(1)}||c_{1-b^{(2)}}^{(2)}|| \cdots ||c_{1-b^{(r)}}^{(r)}||\rho_{b^{(1)}}^{(1)}|| \cdots ||\rho_{b^{(r)}}^{(r)})$
Return $\sigma = (R, \sigma_0, \sigma_1, \sigma_2)$
```

Fig. 7.2: MQDSS-q-n signature generation

In more details, the signer:

- First effectively repeats the $\mathsf{KGen}()$ procedure i.e.,
  - derives $S_{\mathbf{F}}, S_{\mathbf{s}}, S_{\rho}, S_{\mathbf{rte}}$ from $\mathrm{PRG}_{\mathsf{sk}}(\mathsf{sk})$,
  - expands $\mathbf{F} = \mathrm{XOF}_{\mathbf{F}}(S_{\mathbf{F}})$ and $\mathbf{s} = \mathrm{PRG}_{\mathbf{s}}(S_{\mathbf{s}})$ and
  - derives the public key $\mathsf{pk} := (S_{\mathbf{F}}, \mathbf{F}(\mathbf{s}))$.
- Derives a message dependent random value $R = \mathcal{H}(\mathsf{sk} \parallel M)$.
- Using this random value R, the signer computes the randomized message digest $D = \mathcal{H}(R \parallel m)$. The value R must be included in the signature, so that a verifier can derive the same randomized digest.
- Next, the signer uses the pseudorandom generator PRG_{ρ} to generate the values $\rho_0^{(1)}, \ldots, \rho_0^{(r)}, \rho_1^{(1)}, \ldots, \rho_1^{(r)}$ from S_{ρ} and D.
- The signer then uses the pseudorandom generator $\mathrm{PRG}_{\mathbf{rte}}$ to sample the vectors $\mathbf{r}_0^{(1)}, \ldots, \mathbf{r}_0^{(r)}$, $\mathbf{t}_0^{(1)}, \ldots, \mathbf{t}_0^{(r)}$, $\mathbf{e}_0^{(1)}, \ldots, \mathbf{e}_0^{(r)}$ from $S_{\mathbf{rte}}$ and D.

- For each $j \in \{1, \ldots, r\}$
  - Computes $\mathbf{r}_1^{(j)}$ as the difference $\mathbf{s} - \mathbf{r}_0^{(j)}$,
  - Commits to $(\mathbf{r}_0^{(j)}, \mathbf{t}_0^{(j)}, \mathbf{e}_0^{(j)})$ and to $(\mathbf{r}_1^{(j)}, \mathbf{G}(\mathbf{t}_0^{(j)}, \mathbf{r}_1^{(j)}) + \mathbf{e}_0^{(j)})$ applying the commitment functions Com_0 and Com_1, with the random values $\rho_0^{(j)}$ and $\rho_1^{(j)}$ respectively, to obtain $c_0^{(j)}$ and $c_1^{(j)}$ respectively,
  - Sets $\mathsf{com}^{(j)} := (c_0^{(j)}, c_1^{(j)})$.
- Computes the second part of the signature σ_0 as a digest over the concatenation of all commitments $\sigma_0 \leftarrow \mathcal{H}(\mathsf{com}^{(1)}||\mathsf{com}^{(2)}||\ldots||\mathsf{com}^{(r)})$,
- Derives the first challenge $\mathsf{ch}_1 = (\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(r)})$ by applying H_1 to (D, σ_0).
- Using the $\alpha^{(j)}$ as individual challenges per round, the signer computes $\mathbf{t}_1^{(j)} \leftarrow \alpha^{(j)}\mathbf{r}_0^{(j)} - \mathbf{t}_0^{(j)}$ and $\mathbf{e}_1^{(j)} \leftarrow \alpha^{(j)}\mathbf{F}(\mathbf{r}_0^{(j)}) - \mathbf{e}_0^{(j)}$ for all $j \in \{1, \ldots, r\}$,
- Sets the responses $\mathsf{resp}_1^{(j)} := (\mathbf{t}_1^{(j)}, \mathbf{e}_1^{(j)})$ for all $j \in \{1, \ldots, r\}$.
- The concatenation of all responses $\mathsf{resp}_1^{(j)}$ gives the third part of the signature $\sigma_1 \leftarrow (\mathsf{resp}_1^{(1)}||\mathsf{resp}_1^{(2)}||\ldots||\mathsf{resp}_1^{(r)})$.
- The signer computes ch_2 by applying H_2 to the tuple $(D, \sigma_0, \mathsf{ch}_1, \sigma_1)$ and parses it as r binary challenges $b^{(j)} \in \{0, 1\}$.
- For all $j \in \{1, \ldots, r\}$, the signer computes the second responses $\mathsf{resp}_2^{(j)} \leftarrow \mathbf{r}_{b^{(j)}}^{(j)}$.
- Finally, the signer computes the last part of the signature as $\sigma_2 \leftarrow (\mathsf{resp}_2^{(1)}||\mathsf{resp}_2^{(2)}||\ldots||\mathsf{resp}_2^{(r)}||c_{1-b^{(1)}}^{(1)}||c_{1-b^{(2)}}^{(2)}||\ldots||c_{1-b^{(r)}}^{(r)}||\rho_{b^{(1)}}^{(1)}||\ldots||\rho_{b^{(r)}}^{(r)})$, and
- Outputs the signature $\sigma = (R, \sigma_0, \sigma_1, \sigma_2)$.

The complete signature is of the following form:

$$\sigma = (R, \mathcal{H}(\mathsf{com}^{(1)}||\mathsf{com}^{(2)}||\ldots||\mathsf{com}^{(r)}), (\mathsf{resp}_1^{(1)}||\mathsf{resp}_1^{(2)}||\ldots||\mathsf{resp}_1^{(r)}),$$
$$(\mathsf{resp}_2^{(1)}||\mathsf{resp}_2^{(2)}||\ldots||\mathsf{resp}_2^{(r)}||c_{1-b^{(1)}}^{(1)}||c_{1-b^{(2)}}^{(2)}||\ldots||c_{1-b^{(r)}}^{(r)}||\rho_{b^{(1)}}^{(1)}||\ldots||\rho_{b^{(r)}}^{(r)}))$$

As each element of $\mathbb{F}_q$ requires $\lceil \log_2 q \rceil$ bits, and $c_{1-b^{(j)}}^{(j)}$ and $\rho_{b^{(j)}}^{(j)}$ each require $2k$ bits, the signature is $4k + (4k + 3n \lceil \log_2 q \rceil)r$ bits long.

7.4 MQDSS Signature Verification

Upon receiving a message M, a signature $\sigma = (R, \sigma_0, \sigma_1, \sigma_2)$, and a public key $\mathsf{pk} = (S_{\mathbf{F}}, \mathbf{v})$, the verifier performs the verification routine as listed in Figure 7.3.

In more detail, the main goal of the verification process is to reconstruct the missing commitments, and calculate a value σ_0' that will be verified against the inputted σ_0. The whole procedure is as follows:

- Using the pubic key $\mathsf{pk} = (S_{\mathbf{F}}, \mathbf{v})$ and the value R from the signature σ, compute the system parameter $\mathbf{F} \leftarrow \text{XOF}_{\mathbf{F}}(S_{\mathbf{F}})$ and the randomized message digest $D \leftarrow \mathcal{H}(\mathsf{pk}||R||M)$.
- Since the signature contains σ_0, compute the first challenge ch_1 as $\mathsf{ch}_1 \leftarrow H_1(D, \sigma_0)$ and parse it as $ch_1 = (\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(r)})$, $\alpha^{(j)} \in \mathbb{F}_q$
- Next, compute the challenge $\mathsf{ch}_2 \leftarrow H_2(D, \sigma_0, \mathsf{ch}_1, \sigma_1)$, from the two parts σ_0, σ_1 of the signature and the computed ch_1 in the previous step. Parse it as $\mathsf{ch}_2 = (b^{(1)}, b^{(2)}, \ldots, b^{(r)})$, $b^{(j)} \in \{0, 1\}$.

```
$\mathsf{Vf}(\mathsf{pk}, \sigma, M)$

$\mathbf{F} \leftarrow \mathrm{XOF}_{\mathbf{F}}(S_{\mathbf{F}})$
$D \leftarrow \mathcal{H}(\mathsf{pk}||R||M)$
$\mathsf{ch}_1 \leftarrow H_1(D, \sigma_0)$
Parse $\mathsf{ch}_1$ as $\mathsf{ch}_1 = (\alpha^{(1)}, \alpha^{(2)}, \dots, \alpha^{(r)}), \alpha^{(j)} \in \mathbb{F}_q$
$\mathsf{ch}_2 \leftarrow H_2(D, \sigma_0, \mathsf{ch}_1, \sigma_1)$
Parse $\mathsf{ch}_2$ as $\mathsf{ch}_2 = (b^{(1)}, b^{(2)}, \dots, b^{(r)}), b^{(j)} \in \{0, 1\}$
Parse $\sigma_1$ as $\sigma_1 = (\mathsf{resp}_1^{(1)}||\mathsf{resp}_1^{(2)}||\dots||\mathsf{resp}_1^{(r)})$
Parse $\sigma_2$ as $\sigma_2 = (\mathsf{resp}_2^{(1)}||\mathsf{resp}_2^{(2)}||\dots||\mathsf{resp}_2^{(r)}||c_{1-b^{(1)}}^{(1)}||c_{1-b^{(2)}}^{(2)}||\dots||c_{1-b^{(r)}}^{(r)}||\rho_{b^{(1)}}^{(1)}||\dots||\rho_{b^{(r)}}^{(r)})$
For $j \in \{1, \dots, r\}$ do
    Parse $\mathsf{resp}_1^{(j)}$ as $\mathsf{resp}_1^{(j)} = (\mathbf{t}_1^{(j)}, \mathbf{e}_1^{(j)})$
    If $b^{(j)} == 0$
        $\mathbf{r}_0^{(j)} = \mathsf{resp}_2^{(j)}$
        $c_0^{(j)} \leftarrow Com_0(\rho_0^{(j)}, \mathbf{r}_0^{(j)}, \alpha^{(j)}\mathbf{r}_0^{(j)} - \mathbf{t}_1^{(j)}, \alpha^{(j)}\mathbf{F}(\mathbf{r}_0^{(j)}) - \mathbf{e}_1^{(j)})$
    else
        $\mathbf{r}_1^{(j)} = \mathsf{resp}_2^{(j)}$
        $c_1^{(j)} \leftarrow Com_1(\rho_1^{(j)}, \mathbf{r}_1^{(j)}, \alpha^{(j)}(\mathbf{v} - \mathbf{F}(\mathbf{r}_1^{(j)})) - \mathbf{G}(\mathbf{t}_1^{(j)}, \mathbf{r}_1^{(j)}) - \mathbf{e}_1^{(j)})$
    $\mathsf{com}^{(j)} := (c_0^{(j)}, c_1^{(j)})$
$\sigma_0' \leftarrow \mathcal{H}(\mathsf{com}^{(1)}||\mathsf{com}^{(2)}||\dots||\mathsf{com}^{(r)})$
Return $\sigma_0' == \sigma_0$
```

Fig. 7.3: MQDSS-q-n signature verification

- Parse the two signature parts σ_1 and σ_2 as $\sigma_1 = (\mathsf{resp}_1^{(1)}||\mathsf{resp}_1^{(2)}||\dots||\mathsf{resp}_1^{(r)})$ and $\sigma_2 = (\mathsf{resp}_2^{(1)}||\mathsf{resp}_2^{(2)}||\dots||\mathsf{resp}_2^{(r)}||c_{1-b^{(1)}}^{(1)}||c_{1-b^{(2)}}^{(2)}||\dots||c_{1-b^{(r)}}^{(r)}||\rho_{b^{(1)}}^{(1)}||\dots||\rho_{b^{(r)}}^{(r)})$ respectively.

- Since the verifier knows the values $b^{(j)}$ from the previous step, he knows which of the two parts of the commitments $\mathsf{com}^{(j)}$ were included in σ_2, and can now proceed to recovering the other, missing part. This is done for all $j \in \{1, \dots, r\}$ as follows:
  - Parse $\mathsf{resp}_1^{(j)}$ to obtain $(\mathbf{t}_1^{(j)}, \mathbf{e}_1^{(j)})$, and
  - if $b^{(j)} == 0$ compute $c_0^{(j)}$ as $c_0^{(j)} \leftarrow Com_0(\rho_0^{(j)}, \mathbf{r}_0^{(j)}, \alpha^{(j)}\mathbf{r}_0^{(j)} - \mathbf{t}_1^{(j)}, \alpha^{(j)}\mathbf{F}(\mathbf{r}_0^{(j)}) - \mathbf{e}_1^{(j)})$, otherwise compute $c_1^{(j)}$ as $c_1^{(j)} \leftarrow Com_1(\rho_1^{(j)}, \mathbf{r}_1^{(j)}, \alpha^{(j)}(\mathbf{v} - \mathbf{F}(\mathbf{r}_1^{(j)})) - \mathbf{G}(\mathbf{t}_1^{(j)}, \mathbf{r}_1^{(j)}) - \mathbf{e}_1^{(j)})$
  - Set $\mathsf{com}^{(j)} := (c_0^{(j)}, c_1^{(j)})$
- Calculate $\sigma_0' \leftarrow \mathcal{H}(\mathsf{com}^{(1)}||\mathsf{com}^{(2)}||\dots||\mathsf{com}^{(r)})$ from the obtained commitments $\mathsf{com}^{(j)}$
- Return the truth value of $\sigma_0' == \sigma_0$. This means that for verification to succeed, $\sigma_0' = \sigma_0$ should hold.

8

Parameter Sets

8.1 Reference Parameter Sets

Recall (c.f. Section 7.1) that MQDSS-q-n is parameterized by the parameters:

- A positive integer $k \in \mathbb{N}$ - the security parameter,
- A positive integer $n \in \mathbb{N}$ - the number of variables and equations of the system $\mathbf{F}$,
- A positive integer $q \in \mathbb{N}$ (a prime or a prime power) - the order of the finite field $\mathbb{F}_q$,
- A positive integer $r \in \mathbb{N}$ - the number of rounds.

We propose the following two parameter sets as reference parameter sets of MQDSS[1]

- MQDSS-31-48
  - $k = 128$, $q = 31$, $n = 48$, $r = 135$;
- MQDSS-31-64
  - $k = 192$, $q = 31$, $n = 64$, $r = 202$.

The following Table 8.1 summarizes the basic characteristics of these two parameter sets.

	Security category	k	q	n	r	Public key size (bytes)	Secret key size (bytes)	Signature size (bytes)
MQDSS-31-48	1-2	128	31	48	135	46	16	20854
MQDSS-31-64	3-4	192	31	64	202	64	24	43728

Table 8.1: Basic characteristics of the reference parameter sets

We have calculated the key sizes and the signature size based on the formulas provided in Section 7.1: The public key is of length $k + n \lceil \log_2 q \rceil$ bits, the secret key sk of length k bits and the signature of length $4k + (4k + 3n \lceil \log_2 q \rceil)r$ bits[2]. The number of rounds r was calculated as $r = \left\lceil k / \log_2 \frac{2q}{q+1} \right\rceil$.

We also summarize the strength of the reference parameter sets with respect to the best classical and quantum attacks (see Chapter 10). The summary is given in Table 8.2. For more detailed analysis of the algorithms see Chapter 2.

[1] In Version 1.0, for MQDSS-31-48, we had $k = 256$ and $r = 269$; for MQDSS-31-64, we had $k = 384$ and $r = 403$;

[2] In Version 1.0, the signature size was calculated as $2k + (k + 3n \lceil \log_2 q \rceil)r$.

	Best classical attack		Best quantum attack		
	algorithm	Gates	algorithm	Gates	Depth
MQDSS-31-48	HybridF5	2^{160}	Crossbread	2^{99}	2^{90}
MQDSS-31-64	HybridF5	2^{206}	Crossbread	2^{130}	2^{120}

Table 8.2: Best classical and quantum attacks against the reference parameter sets

Remark 8.1. The estimated security of MQDSS-31-64 given in Table 8.2 seems smaller than the required for Security category 3 and 4. However, not all cost is included in the estimate. In particular, the memory requirement for the best attack(s) is huge (the time complexity is only a cube of the memory complexity). This necessarily introduces a huge penalty for memory manipulation. Unfortunately, this is very difficult to estimate, and we have not done it. We hope the community will soon have more insight into this an in general into how to estimate accurately the complexity of memory expensive algorithms.

As part of the submission package we provide reference (and additional) implementations for the two reference parameter sets MQDSS-31-48 and MQDSS-31-64. The details of the implementations are given in Chapter 9 and Chapter 15.

We emphasize that the chosen reference parameter sets are not the only that are suitable for use, and that should be considered by NIST and the broader community in the evaluation process. In the next section we provide additional parameter sets of comparable performance and security strength but over different fields. We decided to keep the original choice of the field as was initially proposed in [16], and to provide implementations only for parameters over $\mathbb{F}_{31}$. We justify our decision in the next section. To match the security levels identified by NIST, we changed the number of rounds in MQDSS-31-64 compared to [16] from 269 to 202 to match Category 3 and 4, and additionally proposed the lower security set MQDSS-31-48 (not in [16]). Furthermore, we decided not to include a parameter set for the Categories 5 and 6 defined by NIST. Over $\mathbb{F}_{31}$ this would be MQDSS-31-88 or even MQDSS-31-96. If NIST shows interest in such a parameter set, it is simple to extend our implementations and performance analysis to this set as well.

We continue the discussion about the additional parameter sets in the next Section.

8.2 Additional Parameter Sets

In addition to MQDSS-31-48 and MQDSS-31-64 we recommend additional parameter sets of comparable security strength i.e. Categories 1-4, as identified by NIST [49], but over different fields. We also provide parameter sets for the much higher security categories 5 and 6. Their basic characteristics and best attacks are given in Tables 8.3 and 8.4. It is important to note that for the additional recommended parameter sets, we do not provide implementation.

It can be noticed that within a security category, the parameter sets over $\mathbb{F}_{16}$ and $\mathbb{F}_{32}$ are of very similar performance characteristics as the reference parameter sets over $\mathbb{F}_{31}$. However, we decided not to include these in the reference parameter sets.

Our decision is based on two main observations: 1. The field $\mathbb{F}_{31}$ is the most natural choice with respect to implementation of the field arithmetic, since *any* platform already contains instructions for multiplication of natural numbers, but no instructions for $\mathbb{F}_{16}$ or

$\mathbb{F}_{32}$. For these fields in general we would have to design specific representations for fast multiplication, or use table lookup instructions. 2. Our optimized implementation using AVX2 instructions is much faster over $\mathbb{F}_{31}$ than $\mathbb{F}_{16}$ or $\mathbb{F}_{32}$. Although, NIST does not require such an implementation (at least at this stage of the standardization process), we believe that in practice it is very relevant.

On the other hand, as $\mathbb{F}_{16}$ and $\mathbb{F}_{32}$ are binary fields, these parameter sets are particularly interesting for hardware implementations. Therefore for hardware, we recommend using them, particularly MQDSS-16-56 and MQDSS-16-72 rather than MQDSS-31-48 and MQDSS-31-64.

Moe generally, in the evaluation process we encourage NIST and the community to treat MQDSS-16-56, MQDSS-16-72, MQDSS-32-48 and MQDSS-32-64 with the same level of attention as the reference parameter sets.

Security category	k	q	n	r	Public key size (bytes)	Secret key size (bytes)	Signature size (bytes)
1-2	128	4	88	189	38	16	24634
1-2	128	16	56	141	44	16	20932
1-2	128	32	48	134	46	16	20700
1-2	128	64	40	131	46	16	20238
3-4	192	4	128	284	56	24	54624
3-4	192	16	72	211	60	24	43140
3-4	192	32	64	201	64	24	43512
3-4	192	64	64	197	72	24	47376
5-6	256	4	160	378	72	32	93872
5-6	256	16	96	281	80	32	76560
5-6	256	31	88	269	87	32	78945
5-6	256	32	88	268	87	32	78652
5-6	256	64	88	262	98	32	85540

Table 8.3: Basic characteristics of the additional parameter sets

Security category			Best classical attack		Best quantum attack		
	q	n	algorithm	Gates	algorithm	Gates	Depth
1-2	4	88	Crossbread	2^{156}	Crossbread	2^{93}	2^{83}
1-2	16	56	HybridF5	2^{171}	Crossbread	2^{98}	2^{89}
1-2	32	48	HybridF5	2^{161}	Crossbread	2^{96}	2^{88}
1-2	64	40	HybridF5	2^{145}	Crossbread	2^{89}	2^{81}
3-4	4	128	Crossbread	2^{230}	Crossbread	2^{129}	2^{119}
3-4	16	72	HybridF5	2^{211}	Crossbread	2^{123}	2^{113}
3-4	32	64	HybridF5	2^{207}	Crossbread	2^{125}	2^{115}
3-4	64	64	HybridF5	2^{220}	Crossbread	2^{136}	2^{127}
5-6	4	160	Crossbread	2^{290}	Crossbread	2^{158}	2^{147}
5-6	16	96	HybridF5	2^{273}	Crossbread	2^{162}	2^{152}
5-6	31	88	HybridF5	2^{275}	Crossbread	2^{179}	2^{168}
5-6	32	88	HybridF5	2^{276}	Crossbread	2^{174}	2^{164}
5-6	64	88	HybridF5	2^{293}	Crossbread	2^{203}	2^{192}

Table 8.4: Best classical and quantum attacks against the additional parameter sets

9

Low Level Description of MQDSS

In Chapter 7, we described the MQDSS scheme in general terms. Here, we complete the specification by giving the byte-level details that should allow an implementer to write a compatible implementation.

This low level description will focus on our reference parameter sets as defined in Section 8.1. Thus, we will assume that $q = 31$, and the underlying field of operation is $\mathbb{F}_{31}$. To provide a slightly more general framework, we will define all functions in terms of parameters $k, n \in \mathbb{N}$, such that $64|k$ and $8|n$. (Such a description will allow application of the following detailed specifications not only to the reference parameter sets, but also to parameter sets over $\mathbb{F}_{31}$ of different security level.)

9.1 Auxiliary Functions

9.1.1 Secret Key Expansion

The secret key of MQDSS, denoted by sk is a $k/8$-byte string. It is used in the key generation process (see Section 7.2) and in the signing process (see Section 7.3) where it is first expanded to four separate values: $S_{\mathbf{F}}$, $S_{\mathbf{s}}$, S_ρ and $S_{\mathbf{rte}}$. This is done by expanding to $k/2$ bytes, and interpreting the first $k/8$ bytes as $S_{\mathbf{F}}$, the second $k/8$ as $S_{\mathbf{s}}$, the third $k/8$ as S_ρ and the last $k/8$ as $S_{\mathbf{rte}}$ (see Algorithm 1).

Algorithm 1 SecretKeyExpansion(sk)

```
Input: $\mathsf{sk}$
    $block \leftarrow \text{SHAKE256}(\mathsf{sk}, 4k/8)$
    $S_{\mathbf{F}} \leftarrow \text{subarray}(block, 0, k/8)$
    $S_{\mathbf{s}} \leftarrow \text{subarray}(block, k/8, 2k/8)$
    $S_\rho \leftarrow \text{subarray}(block, 2k/8, 3k/8)$
    $S_{\mathbf{rte}} \leftarrow \text{subarray}(block, 3k/8, 4k/8)$
Output: $S_{\mathbf{F}}$, $S_{\mathbf{s}}$, $S_\rho$ $S_{\mathbf{rte}}$
```

9.1.2 Expanding $S_{\mathbf{F}}$, $S_{\mathbf{s}}$ and $S_{\mathbf{rte}}$

The functions $\text{XOF}_{\mathbf{F}}$, $\text{PRG}_{\mathbf{s}}$ and $\text{PRG}_{\mathbf{rte}}$ are instantiated using rejection sampling of the output of the extendable output function SHAKE256 standardized in FIPS 202, the SHA-3 standard [48]. The rejection sampling works as follows: For each output byte of SHAKE256, we ignore the most significant three bits. We discard the resulting value if it is equal to 31 when interpreted as an unsigned integer (i.e. all five bits are set). See Algorithm 2 for details.

Algorithm 2 RejectSample($seed, len$)

```
Input: $seed$, $len$
  $array31 \leftarrow [\,]$
  $state \leftarrow$ SHAKE256absorb($seed$)
  while len($array31$) $< len$ do
      $block \leftarrow$ SHAKE256squeeze($state$)
      $i \leftarrow 0$
      while $i <$len($block$) $\wedge$ len($array31$) $< len$ do
          $cand \leftarrow block[i]$
          if trunc($cand, 5$) $<>$ 11111 then
              append($array31$,mask($cand, 5, 7$))
          end if
          $i \leftarrow i + 1$
      end while
  end while
Output: $array31$
```

Using Algorithm 2 we can easily expand all the necessary values. **F** is obtained by direct application of the RejectSample algorithm. The output elements are then interpreted as integers. The elements of **F** are in signed integers between -15 and 15, inclusive. We bring the randomly sampled integer to this domain by subtracting 15. Algorithm 3 is our wrapper for this function.

Algorithm 3 $\mathcal{MQ}$system($S_{\mathbf{F}}$)

```
Input: $S_{\mathbf{F}}$
  $\mathbf{F} \leftarrow$RejectSample($S_{\mathbf{F}}, n(\frac{n(n+1)}{2} + n)$)
  for $0 \leqslant i <$len($\mathbf{F}$) do
      $\mathbf{F}[i] \leftarrow \mathbf{F}[i] - 15$
  end for
Output: $\mathbf{F}$
```

The secret vector **s** is derived similarly, with the crucial difference that the secret key elements are not transformed to signed integers. The random elements for the vectors $\mathbf{r}_0$, $\mathbf{t}_0$ and $\mathbf{e}_0$ are derived from the seed $S_{\mathbf{rte}}$ in exactly the same way as the secret vector. Note that we derive all vectors of the same type for all rounds consecutively, i.e. $\mathbf{r}_0^{(1)}$, $\mathbf{r}_0^{(2)}$, $\mathbf{r}_0^{(3)}$, $\dots$, $\mathbf{t}_0^{(1)}$, $\mathbf{t}_0^{(2)}$, $\dots$, $\mathbf{e}_0^{(1)}$, $\mathbf{e}_0^{(2)}$, $\dots$ rather than $\mathbf{r}_0^{(1)}$, $\mathbf{t}_0^{(1)}$, $\mathbf{e}_0^{(1)}$, $\mathbf{r}_0^{(2)}$, $\mathbf{t}_0^{(2)}$, $\mathbf{e}_0^{(3)}$, $\dots$. (See Algorithm 4 and Algorithm 5).

Algorithm 4 SecretVector($S_{\mathbf{s}}$)

```
Input: $S_{\mathbf{s}}$
  $\mathbf{s} \leftarrow$RejectSample($S_{\mathbf{F}}, n$)
Output: $\mathbf{s}$
```

9.1.3 Evaluating F

At the core of the scheme lies evaluation of the **F** function, and its bilinear counterpart **G**. The evaluation of **F** can roughly be divided in two parts: the generation of all quadratic terms, and computation of the resulting polynomials for given terms.

For the generation of the quadratic terms, we construct the terms in a variant of graded reverse lexicographic order. We note that for most platforms, this is not the most

Algorithm 5 RTEexpand($S_{\mathbf{rte}}, D$)

```
Input: $S_{\mathbf{rte}}$
  $array_{\mathbf{rte}}$ ←RejectSample($S_{\mathbf{rte}}||D, 3rn$)
  $array_{\mathbf{r}}$ ←subarray($array_{\mathbf{rte}}, 0, rn$)
  $array_{\mathbf{t}}$ ←subarray($array_{\mathbf{rte}}, rn, 2rn$)
  $array_{\mathbf{e}}$ ←subarray($array_{\mathbf{rte}}, 2rn, 3rn$)
  $\mathbf{r} \leftarrow [], \mathbf{t} \leftarrow [], \mathbf{e} \leftarrow []$
  for $0 \leqslant i < rn; i \leftarrow i + n$ do
    $append(\mathbf{r}, subarray(array_{\mathbf{r}}, i, i + n))$
    $append(\mathbf{t}, subarray(array_{\mathbf{t}}, i, i + n))$
    $append(\mathbf{e}, subarray(array_{\mathbf{e}}, i, i + n))$
  end for
Output: $\mathbf{r}$, $\mathbf{t}$, $\mathbf{e}$
```

efficient way to generate the quadratic terms [16], but it provides a reasonably straightforward method that has decent average performance. Adhering to the same order is crucial for implementations to be compatible, as this determines which elements of the system parameter coincide with which terms.

In order to somewhat accommodate platforms that have combined multiplication and addition instructions, (e.g. the `vpmaddubs` instruction on AVX2), we process pairs of quadratic terms rather than individual coefficients. This format is chosen to still be convenient to handle on platforms that cannot combine multiplications and additions. In particular, platforms with more traditional SIMD instructions can use unpack instructions to de-interleave the vectors.

Algorithm 6 EvaluateF($\mathbf{u}, \mathbf{F}$)

```
Input: $\mathbf{u}, \mathbf{F}$
  $terms \leftarrow [\ ]$
  for $0 \leqslant i < n$ do
    for $0 \leqslant j < i$ do
      append($terms, \mathbf{u}[i] \cdot \mathbf{u}[j] \mod 31$)
    end for
  end for
  $\mathbf{r} \leftarrow [0|j = 0..n - 1]$
  for $0 \leqslant i < n;\ i \leftarrow i + 2$ do
    for $0 \leqslant j < n$ do
      $\mathbf{r}[j] \leftarrow \mathbf{r}[j] + \mathbf{u}[i] \cdot \mathbf{F}[i \cdot n + 2 \cdot j] \mod 31$
      $\mathbf{r}[j] \leftarrow \mathbf{r}[j] + \mathbf{u}[i+1] \cdot \mathbf{F}[i \cdot n + 2 \cdot j + 1] \mod 31$
    end for
  end for
  for $0 \leqslant i < \frac{n \cdot (n+1)}{2};\ i \leftarrow i + 2$ do
    for $0 \leqslant j < n$ do
      $\mathbf{r}[j] \leftarrow \mathbf{r}[j] + \text{terms}[i] \cdot \mathbf{F}[n \cdot m + i \cdot m + 2 \cdot j] \mod 31$
      $\mathbf{r}[j] \leftarrow \mathbf{r}[j] + \text{terms}[i+1] \cdot \mathbf{F}[n \cdot m + i \cdot m + 2 \cdot j + 1] \mod 31$
    end for
  end for
Output: $\mathbf{r} = \mathbf{F}(\mathbf{u})$
```

We describe the process in pseudo-code below, see Algorithm 6. Note that this includes multiplication with elements in $\mathbf{F}$ over $\mathbb{F}_{31}$.

To evaluate the polar form function $\mathbf{G}$, we use almost the same procedure. For completeness, we list it in Algorithm 7. Notably, the differences are limited to a different term generation, and skipping of the square terms (as these cancel out).

Algorithm 7 EvaluateG($\mathbf{u}, \mathbf{v}, \mathbf{F}$)

```
Input: $\mathbf{u}, \mathbf{v}, \mathbf{F}$
  $terms \leftarrow [\,]$
  for $0 \leqslant i < n$ do
      for $0 \leqslant j < i$ do
          append($terms, \mathbf{u}[i] \cdot \mathbf{v}[j] + \mathbf{u}[j] \cdot \mathbf{v}[i] \mod 31$)
      end for
  end for
  $\mathbf{r} \leftarrow [0 | j = 1..n]$
  for $0 \leqslant i < \frac{n \cdot (n+1)}{2}$; $i \leftarrow i + 2$ do
      for $0 \leqslant j < n$ do
          $\mathbf{r}[j] \leftarrow \mathbf{r}[j] + \text{terms}[i] \cdot \mathbf{F}[n \cdot m + i \cdot m + 2 \cdot j] \mod 31$
          $\mathbf{r}[j] \leftarrow \mathbf{r}[j] + \text{terms}[i+1] \cdot \mathbf{F}[n \cdot m + i \cdot m + 2 \cdot j + 1] \mod 31$
      end for
  end for
Output: $\mathbf{r} = \mathbf{G}(\mathbf{u}, \mathbf{v})$
```

9.1.4 Packing and unpacking $\mathbb{F}_{31}$ elements

All field elements included in the signature are stored in packed representation. This means that, when storing a vector of $\mathbb{F}_{31}$ elements, each element is expressed using five bits, representing the element using its smallest non-negative representation as an integer. The first byte of the byte sequence represents the first five bits of the first element, and the three least-significant bits of the next element. The next byte contains the remaining two high bits of the second element, the complete third element, and the least-significant bit of the forth element, etc. Note that for all parameters of MQDSS, we have restricted the value of n to be a multiple of 8. Thus, there is no need to explicitly specify padding, since this will result in byte arrays of exact multiples of eight bits. A vector of elements in $\mathbb{F}_{31}$ is unpacked by applying the inverse of the above operation (see Algorithms 8 and 9).

Algorithm 8 PackArray31($\mathbf{u}$)

```
Input: $\mathbf{u}$
  $bitstring \leftarrow [\,]$, $bytearray \leftarrow [\,]$
  for $0 \leqslant i < len(\mathbf{u})$ do
      $bitstring \leftarrow bitstring || \text{trunc}(\mathbf{u}[\text{i}], 5)$
  end for
  for $0 \leqslant i <$len($bitstring$); $i \leftarrow i + 8$ do
      append($bytearray$,subarray($bitstring, i, i + 8$))
  end for
Output: $bytearray$
```

Algorithm 9 UnpackArray31($bytearray$)

```
Input: $bytearray$
  $bitstring \leftarrow [\,]$, $\mathbf{u} \leftarrow [\,]$
  for $0 \leqslant i < len(bytearray)$ do
      $bitstring \leftarrow bitstring || bytearray[i]$
  end for
  for $0 \leqslant i <$len($bitstring$); $i \leftarrow i + 5$ do
      append($\mathbf{u}$,subarray($bitstring, i, i + 5$)||000)
  end for
Output: $\mathbf{u}$
```

9.1.5 Expanding S_ρ

The function PRG_ρ is instantiated using SHAKE256. See Algorithm 10.

Algorithm 10 RhoExpand(S_ρ, D)

```
Input: $S_\rho$, $D$
  $block \leftarrow \mathrm{SHAKE256}(S_\rho || D,\ r \cdot 4k/8)$
  $\rho_0 \leftarrow \mathrm{subarray}(block, 0, r \cdot 2k/8)$
  $\rho_1 \leftarrow \mathrm{subarray}(block, r \cdot 2k/8, r \cdot 4k/8)$
Output: $\rho_0$, $\rho_1$
```

9.1.6 Commitment and hash functions

The commitments Com_0 and Com_1 and the Hash functions $\mathcal{H}$, H_1 and H_2 are instantiated also using SHAKE256. They take as input arrays of $\mathbb{F}_{31}$ elements that need to be in packed form. The input to the commitment functions Com_0 and Com_1 is a random string followed by a sequence of three, respectively two packed byte arrays. The arrays are simply concatenated bytewise, starting with the vector listed first. The same applies for the hash functions $\mathcal{H}$, H_1 and H_2. Their algorithmic description is given in Algorithms 11-15.

It should come as no surprise that the same rejection sampling method is applied to sample the challenges $\alpha^{(i)} \in \mathbb{F}_{31}$. After absorbing the transcript into the SHAKE state, it is repeatedly squeezed until sufficient elements have been extracted – as before, the least significant 5 bits are considered as an unsigned integer, and is rejected if it is equal to 31.

The binary challenges are obtained by enumerating the bits of the hash output per byte, from least to most significant. (see Algorithm 15.)

Algorithm 11 Com0($\rho_0, \mathbf{r}, \mathbf{t}, \mathbf{e}$)

```
Input: $\mathbf{r}, \mathbf{t}, \mathbf{e}$
  $c_0 \leftarrow [\,]$
  $seed \leftarrow (\rho_0, \mathrm{PackArray31}(\mathbf{r}) || \mathrm{PackArray31}(\mathbf{t}) || \mathrm{PackArray31}(\mathbf{e}))$
  $state \leftarrow \mathrm{SHAKE256absorb}(seed)$
  $block \leftarrow \mathrm{SHAKE256squeeze}(state)$
  $c_0 \leftarrow \mathrm{subarray}(block, 0, 2k/8)$
Output: $c_0$
```

Algorithm 12 Com1($\rho_1, \mathbf{r}, \mathbf{e}$)

```
Input: $\mathbf{r}, \mathbf{e}$
  $c_1 \leftarrow [\,]$
  $seed \leftarrow (\rho_1, \mathrm{PackArray31}(\mathbf{r}) || \mathrm{PackArray31}(\mathbf{e}))$
  $state \leftarrow \mathrm{SHAKE256absorb}(seed)$
  $block \leftarrow \mathrm{SHAKE256squeeze}(state)$
  $c_1 \leftarrow \mathrm{subarray}(block, 0, 2k/8)$
Output: $c_1$
```

Algorithm 13 Hash($bytearray$)

```
Input: $bytearray$
  $state \leftarrow$ SHAKE256absorb($bytearray$)
  $block \leftarrow$ SHAKE256squeeze($state$)
  $digest \leftarrow$ subarray($block, 0, 2k/8$)
Output: $digest$
```

Algorithm 14 Hash1(D, σ_0)

```
Input: $D, \sigma_0$
  $seed \leftarrow D||\sigma_0$
  $\mathsf{ch}_1 \leftarrow$RejectSample[$seed, r$]
Output: $\mathsf{ch}_1$
```

Algorithm 15 Hash2($D, \sigma_0, \mathsf{ch}_1, \sigma_1$)

```
Input: $D, \sigma_0, \mathsf{ch}_1, \sigma_1$
  $seed \leftarrow D||\sigma_0$PackArray31($\mathsf{ch}_1$)$||\sigma_1$
  $state \leftarrow$ SHAKE256absorb($seed$)
  $block \leftarrow$ SHAKE256squeeze($state$)
  $\mathsf{ch}_2 \leftarrow$[ ]
  for $0 \leqslant i < r$ do
    $temp = block$[floor($i/8$)]
    append($\mathsf{ch}_2, temp[i \mod 8]$)
  end for
Output: $\mathsf{ch}_2$
```

9.2 Putting it all together - Pseudo code of KGen,Sign,Vf

Using the defined auxiliary functions from the previous section, we can provide a low level algorithmic description of the defining algorithms of MQDSS - KGen,Sign,Vf (see Chapter 7).

For the KGen algorithm of MQDSS, we assume the existence of a function rand() that on input $k \in \mathbb{N}$ outputs $k/8$ bytes of strong randomness. It is given in Algorithm 16.

Algorithm 16 KGen(k)

```
Input: $k$
  $\mathsf{sk} \leftarrow$rand($k$)
  $S_{\mathbf{F}}, S_\rho, S_{\mathbf{s}}, S_{\mathbf{rte}} \leftarrow$ SecretKeyExpansion($\mathsf{sk}$)
  $\mathbf{F} \leftarrow\mathcal{MQ}$system($S_{\mathbf{F}}$)
  $\mathbf{s} \leftarrow$SecretVector($S_{\mathbf{s}}$)
  $\mathbf{v} \leftarrow$EvaluateF($\mathbf{s}, \mathbf{F}$)
  $\mathsf{pk} \leftarrow S_{\mathbf{F}}||$PackArray31($\mathbf{v}$)
Output: $(\mathsf{pk}, \mathsf{sk})$
```

An MQDSS signature is generated with the algorithm Sign (see Algorithm 17). It takes as input a secret key sk and a message to be signed M.

An MQDSS signature is verified using the algorithm Vf (see Algorithm 18). It takes as input a public key sk, a message M, and a signature σ.

Algorithm 17 $\mathsf{Sign}(\mathsf{sk}, M)$

```
Input: $\mathsf{sk}, M$
  $S_{\mathbf{F}}$, $S_{\mathbf{s}}$, $S_\rho$, $S_{\mathbf{rte}} \leftarrow$ SecretKeyExpansion($\mathsf{sk}$)
  $\mathbf{F} \leftarrow \mathcal{MQ}$ system($S_{\mathbf{F}}$)
  $\mathbf{s} \leftarrow$SecretVector($S_{\mathbf{s}}$)
  $\mathbf{v} \leftarrow$EvaluateF($\mathbf{s}, \mathbf{F}$)
  $\mathsf{pk} \leftarrow S_{\mathbf{F}}||$PackArray31($\mathbf{v}$)
  $R \leftarrow$Hash($\mathsf{sk}||M$)
  $D \leftarrow$Hash($\mathsf{pk}||R||M$)
  $\rho_0, \rho_1 \leftarrow$ RhoExpand($S_\rho, D$)
  $\mathbf{r}_0, \mathbf{t}_0, \mathbf{e}_0 \leftarrow$ RTEexpand($S_{\mathbf{rte}}, D$)
  $\mathbf{r}_1 \leftarrow [\,]$, $\mathbf{t}_1 \leftarrow [\,]$, $\mathbf{e}_1 \leftarrow [\,]$
  $c_0 \leftarrow [\,]$, $c_1 \leftarrow [\,]$
  $\mathsf{com} \leftarrow [\,]$
  for $0 \leqslant i < r$ do
      append($\mathbf{r}_1, \mathbf{s} - \mathbf{r}_0[i]$)
      append($c_0$,Com0($\rho_0[i], \mathbf{r}_0[i], \mathbf{t}_0[i], \mathbf{e}_0[i]$))
      append($c_1$,Com1($\rho_1[i], \mathbf{r}_1[i]$,EvaluateG($\mathbf{t}_0[i], \mathbf{r}_1[i], \mathbf{F}$) $+ \mathbf{e}_0[i]$))
      $\mathsf{com} \leftarrow \mathsf{com}||$PackArray31($c_0[i]$)$||$PackArray31($c_1[i]$)
  end for
  $\sigma_0 \leftarrow$Hash($\mathsf{com}$)
  $\mathsf{ch}_1 \leftarrow$Hash1($D, \sigma_0$)
  $\sigma_1 \leftarrow [\,]$
  for $0 \leqslant i < r$ do
      $\mathbf{t}_1[i] \leftarrow \mathsf{ch}_1[i] \cdot \mathbf{r}_0[i] - \mathbf{t}_0[i]$
      $\mathbf{e}_1[i] \leftarrow \mathsf{ch}_1[i] \cdot$EvaluateF($\mathbf{r}_0[i], \mathbf{F}$)$- \mathbf{e}_0[i]$
      $\sigma_1 \leftarrow \sigma_1||$PackArray31($\mathbf{t}_1[i]$)$||$PackArray31($\mathbf{e}_1[i]$)
  end for
  $\mathsf{ch}_2 \leftarrow$Hash2($D, \sigma_0, \mathsf{ch}_1, \sigma_1$)
  $\sigma_2 \leftarrow [\,]$
  for $0 \leqslant i < r$ do
      if $\mathsf{ch}_2[i] == 0$ then
          $\sigma_2 \leftarrow \sigma_2||$PackArray31($\mathbf{r}_0[i]$)
      else
          $\sigma_2 \leftarrow \sigma_2||$PackArray31($\mathbf{r}_1[i]$)
      end if
  end for
  for $0 \leqslant i < r$ do
      if $\mathsf{ch}_2[i] == 0$ then
          $\sigma_2 \leftarrow \sigma_2||c_1[i]$
      else
          $\sigma_2 \leftarrow \sigma_2||c_0[i]$
      end if
  end for
  for $0 \leqslant i < r$ do
      if $\mathsf{ch}_2[i] == 0$ then
          $\sigma_2 \leftarrow \sigma_2||\rho_0[i]$
      else
          $\sigma_2 \leftarrow \sigma_2||\rho_1[i]$
      end if
  end for
Output: $\sigma = R||\sigma_0||\sigma_1||\sigma_2$
```

Algorithm 18 $\mathsf{Vf}(\mathsf{pk}, \sigma, M)$

```
Input: $\mathsf{pk}, \sigma, M$
  $R \leftarrow \text{subarray}(\sigma, 0, 2k/8)$
  $\sigma_0 \leftarrow \text{subarray}(\sigma, 2k/8, 4k/8)$
  $\sigma_1 \leftarrow \text{subarray}(\sigma, 4k/8, (4k + 10nr)/8)$
  $\sigma_2 \leftarrow \text{subarray}(\sigma, (4k + 10nr)/8, len(\sigma))$
  $S_{\mathbf{F}} \leftarrow \text{subarray}(\mathsf{pk}, 0, k/8)$
  $\mathbf{F} \leftarrow \mathcal{MQ}\,\text{system}(S_{\mathbf{F}})$
  $D \leftarrow \text{Hash}(\mathsf{pk}||R||M)$
  $\mathsf{ch}_1 \leftarrow \text{Hash1}(D, \sigma_0)$
  $\mathsf{ch}_2 \leftarrow \text{Hash2}(D, \sigma_0, \mathsf{ch}_1, \sigma_1)$
  $\mathsf{resp}_1 \leftarrow \text{UnpackArray31}(\sigma_1)$
  $\mathsf{resp}_2 \leftarrow \text{UnpackArray31}(\text{subarray}(\sigma_2, 0, 5nr/8))$
  $c \leftarrow \text{subarray}(\sigma_2, 5nr/8, 5nr/8 + 2kr/8)$
  $\rho \leftarrow \text{subarray}(\sigma_2, 5nr/8 + 2kr/8, len(\sigma_2))$
  $\mathsf{com} \leftarrow [\,]$
  for $0 \leqslant i < r$ do
    $\mathbf{t}_1 \leftarrow \mathsf{resp}_1[2i]$
    $\mathbf{e}_1 \leftarrow \mathsf{resp}_1[2i + 1]$
    if $\mathsf{ch}_2[i] == 0$ then
      $\mathbf{r}_0 \leftarrow \mathsf{resp}_2[i]$
      $c_0 \leftarrow \text{Com0}(\rho[i], \mathbf{r}_0, \mathsf{ch}_1[i] \cdot \mathbf{r}_0 - \mathbf{t}_1, \mathsf{ch}_1[i] \cdot \text{EvaluateF}(\mathbf{r}_0, \mathbf{F}) - \mathbf{e}_1)$
      $c_1 \leftarrow \text{subarray}(c, i \cdot k/8, (i + 1) \cdot k/8)$
    else
      $\mathbf{r}_1 \leftarrow \mathsf{resp}_2[i]$
      $c_1 \leftarrow \text{Com1}(\rho[i], \mathbf{r}_1, \mathsf{ch}_1[i] \cdot (\mathbf{v} - \text{EvaluateF}(\mathbf{r}_1, \mathbf{F})) - \text{EvaluateG}(\mathbf{t}_1, \mathbf{r}_1, \mathbf{F}) - \mathbf{e}_1)$
      $c_0 \leftarrow \text{subarray}(c, i \cdot k/8, (i + 1) \cdot k/8)$
      $\mathsf{com} \leftarrow \mathsf{com}||c_0||c_1$
    end if
  end for
  $\sigma_0' \leftarrow \text{Hash}(\mathsf{com})$
Output: $\sigma_0' == \sigma_0$
```

10

Security of MQDSS

10.1 EU-CMA security of MQDSS

The security of MQDSS was proven in [16]. The security reduction is in the random oracle model and builds on the results obtained for $q2$ signature schemes (see Appendix A, Section A.1). For completeness we provide the full proof in Appendix A, Section A.2.

Theorem 10.1. MQDSS *is EU-CMA-secure in the random oracle model, if the following conditions are satisfied:*

- *the search version of the $\mathcal{MQ}$ problem is intractable in the average case,*
- *the hash functions $\mathcal{H}$, H_1, and H_2 are modeled as random oracles,*
- *the commitment functions Com_0 and Com_1 are computationally binding, computationally hiding, and have $\mathcal{O}(k)$ bits of output entropy,*
- *the function XOF_F is modeled as random oracle and*
- *the pseudorandom generators PRG_{sk}, $PRG_{\mathbf{s}}$, PRG_{ρ} and $PRG_{\mathbf{rte}}$ have outputs computationally indistinguishable from random for any polynomial time adversary.*

MQDSS instantiates the commitments Com_0 and Com_1 using cryptographically secure hash function (in particular SHAKE256). The following theorem from [45] shows that modeling the hash function as a random oracle, we can show that we obtain a computationally hiding commitment. Furthermore, the theorem gives a lower bound on the randomness that needs to be included as input to the random oracle.

Theorem 10.2. *Let k be the security parameter. Let $H : \{0,1\}^* \to \{0,1\}^n$ be a random oracle. Let $\rho \in \{0,1\}^{2k}$ be uniformly random. Then the commitment $H(\rho, m)$ is computationally hiding. In particular, for any two messages M, M' the advantage of a polynomial time (quantum) adversary*

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A}\left(\mathsf{Com}(\rho, M)\right)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A}\left(\mathsf{Com}(\rho, M')\right)\right]\right| \leqslant L(q) \cdot 1/2^k.$$

where $L(q)$ is some linear function of the total number of queries of $\mathcal{A}$.

10.2 Attacks Against MQDSS

Having shown the EU-CMA security of MQDSS, the best attacks against the cryptosystem are against the conditions that provide the security. Thus, an adversary could:

- Attack the $\mathcal{MQ}$ problem,
- Attack the computationally binding property of the commitments
- Attack the computationally hiding property of the commitments
- Attack the hash functions
- Attack the pseudo-random generators

Since the commitment functions, the hash functions and the pseudo-random generators are all instantiated using SHAKE256, all the attacks apart from the first boil down to attacking SHAKE256.

One could compromise the security of MQDSS if one breaks the preimage resistance (this will break the hiding property of the commitments), the collision resistance (this will break the binding property of the commitments) or if one finds properties that distinguish the output of SHAKE256 from random. A substantial amount of research has been devoted to the security of SHAKE and the SHA3 standard. The public scrutiny gives confidence in its security, however the details are out of the scope of this document. We refer the interested reader to [48, 11].

This leaves attacks against the $\mathcal{MQ}$ problem as the point of interest. Since the public key is randomly generated (from a random seed by expanding the seed), the obtained system can be considered as semiregular, i.e. we can be confident that there are no hidden structural weaknesses. This means that the generic algebraic methods are the best algorithms against the $\mathcal{MQ}$ instance in MQDSS and therefore against the system. For details see 2. Based on these conclusions, the security of the proposed parameter sets can be estimated as in Table 8.2. Additional parameters security estimate is given in Table 8.4, and scaled-down parameters estimate in Table 13.3.

11

Design Rationale

In this chapter we discuss all relevant design choices that we made and provide appropriate justification for these choices.

11.1 Parameters

In choosing appropriate parameters for MQDSS, the most important criteria was of course the level of security these parameters provide. In the previous chapters we provided a complete security analysis of MQDSS. We

- proved the security of MQDSS in the random oracle model (cf. Section 10.1),
- analyzed the practical security of the $\mathcal{MQ}$ problem by investigating the state of the art classical and quantum algorithms for solving it (c.f. Section 2.2 and Section 2.3), and
- used known results about the security of the extendable output function SHAKE256, which we used to instantiate the commitments, the pseudo-random generators and hash functions.

Since our security reduction in the ROM is very loose, we found it impractical to use concrete expressions from the reduction in our choice of parameters. Instead, the parameters are based on the best known attacks against the $\mathcal{MQ}$ problem and against SHAKE256. In particular,

- We choose the number of variables and equations in $\mathbf{F}$ to be the same i.e $m = n$, as this gives effectively the hardest instances of the $\mathcal{MQ}$ problem.
- Using the analysis from Section 2.2, we estimate the lower bound of the number of variables n' in order for the resulting $\mathcal{MQ}$ instance to satisfy a particular security level/category (as defined in [49]) in terms of classical field operations of the best classical attacks,
- Using the analysis from Section 2.3, we estimate the lower bound of the number of variables n'' in order for the resulting $\mathcal{MQ}$ instance to satisfy a particular security level/category (as defined in [49]) in terms of quantum circuit size and depth of the best quantum attacks,
- The number of variables n is then chosen as $n = \max\{n', n''\}$.
- We choose the parameter k such that the output of the hash functions $\mathcal{H}, H_1, H_2$ is large enough to satisfy collision resistance security of the level specified by Categories 2,4 and 6.

- We chose the parameter ρ denoting the input randomness to the commitment functions to be $\rho = 2k$ based on the results from [45] (See Theorem 10.2)[1].
- Finally, the number of rounds r is chosen such that the parallel composition of r rounds of the SSH 5-pass IDS has soundness error $< 1/2^k$.

11.2 5-pass over 3-pass SSH Identification Scheme

In [43], Sakumoto, Shirai and Hiwatari propose also a 3-pass scheme whose security also provably relies on the $\mathcal{MQ}$ problem and is defined solely over $\mathbb{F}_2$. One could argue that this one is a much more natural choice. Indeed, it is a 3-pass scheme, so one can directly apply the Fiat-Shamir transform that has been scrutinized for decades by the community. In addition it is defined over the Boolean domain, so implementation is particularly easy. However, a careful analysis shows that it has a serious drawback, that make it clearly inferior compared to the 5-pass SSH Identification scheme.

The 3-pass SSH scheme has a soundness error of 2/3 which is greater than $\frac{q+1}{2q}$ (the 5-pass SSH soundness error) for any $q > 2$. Thus for example, in order to satisfy Categories 1-2, the number of rounds would have to be 219, which is much larger than 135 - the number of rounds in MQDSS-31-48 (Security categories 1-2). Now, for Categories 1-2, the number of variables in the $\mathcal{MQ}$ system needs to be at least $n = 160$, which amounts to a signature of size $4k + r(3n + 4k) = 27220$ bytes (see [16] for derivation of this formula) which is more than 6.2KiB larger than MQDSS-31-48. For Categories 3-4 the difference is even larger - almost 15.2KiB, since we now need at least $r = 329$ and $n = 224$ which gives a signature of size 59316 bytes.

11.3 Optimizations

In the definition of MQDSS (see Chapter 7) we have used an optimization proposed already in [43]: It is not necessary to include all $2r$ commitments in the transcript. Instead, we include a digest over the concatenation of all commitments $\sigma_0 = \mathcal{H}(\mathsf{com}^{(1)}||\mathsf{com}^{(2)}||\ldots||\mathsf{com}^{(r)})$ and also the commitments $c^{(1)}_{1-b^{(1)}}, c^{(2)}_{1-b^{(2)}}, \ldots, c^{(r)}_{1-b^{(r)}}$ that the verifier can not recompute. This optimization saves $4kr - 2k$ bits from the final signature which is more than 8.4KiB for MQDSS-31-48 and more than 18.9KiB for MQDSS-31-64. This modification does not cause any problems, since we have shown (c.f.Chapter 10) that it does not disturb the security arguments.

11.4 Other Functions

In order to instantiate the commitment functions, pseudorandom generators and extendable output function, we rely on SHAKE-256, as standardized in FIPS 202, the SHA-3 standard. This gives us a sufficiently large security margin that its preimage and second preimage resistance is not relevant for the overall security level. In general, this means that we simply concatenate the defined inputs as byte arrays, and absorb them into the SHAKE state. Chapter 9 provides some more detail on specific usage.

[1] In Version 1, we wrongfully assumed that the randomness of the other inputs to the commitment is enough to provide the computationally hiding property, and used $\rho = 0$. This mistake is corrected in this version, although, unfortunately it results in some overhead in the signature size.

12

Performance Analysis

12.1 Performance on Intel x64-86

In order to obtain benchmarks, we evaluate our reference implementation on a machine using the Intel x64-86 instruction set. In particular, we use a single core of a 3.5 GHz Intel Core i7-4770K CPU. We follow the standard practice of disabling TurboBoost and hyper-threading. The system has 32 KiB L1 instruction cache, 32 KiB L1 data cache, 256 KiB L2 cache and 8192 KiB L3 cache. Furthermore, it has 32GiB of RAM, running at 1333 MHz. When performing the benchmarks, the system ran on Linux kernel 4.9.0-4-amd64, Debian 9 (Stretch).

We compiled the code using GCC version `6.3.0-18`, with the compiler optimization flag `-O3`. The median resulting cycle counts are listed in the table below.

	keygen	signing	verification
MQDSS-31-48	1 192 984	26 630 590	19 840 136
MQDSS-31-64	2 767 384	85 268 712	62 306 098

12.2 Performance on Intel x64-86 AVX2

Since the evaluation of the MQ function is the most costly part of the computation but also benefits greatly from parallelism, we thought it useful to also provide benchmarks when the scheme is implemented using AVX2 instructions. We used the same system described above to obtain the following measurements, this time including the `-mavx2` compiler flag.

	keygen	signing	verification
MQDSS-31-48	1 074 644	3 816 106	2 551 270
MQDSS-31-64	2 491 050	9 047 148	6 132 948

12.3 Size

As the private key is merely a seed that is used to generate the required secret material, this is 16 respectively 24 bytes for the given parameter sets. The public key contains a public seed, but also $\mathbf{F}(\mathbf{s})$, making it 46 and 64 bytes respectively.

The stack space consumption is largely determined by the size of the signature and the expanded version of $\mathbf{F}$. A straight-forward implementation constructs the transcript in memory before evaluating the hash function that determines the challenges. More memory-conservative implementations could keep an intermediate hash function state, instead, and stream through the transcript as it is constructed.

The expanded version of $\mathbf{F}$ requires some active memory. Naively, it benefits from having 57 KiB or 100 KiB (for the different parameter sets, respectively) of active memory available. More memory-constrained implementations could reschedule the different computations in a way that $\mathbf{F}$ only needs to be parsed once, however, and can thus also make use of a streaming API.

For the given parameter sets, the signature size is respectively 20854 and 43728 bytes (i.e. 20 KiB and 42 KiB). Since the signature primarily consists of transcripts of rounds of the non-interactive identification protocol, it scales linearly in the number of rounds and in the size of the vectors (see Chapter 5 for more details on this).

13

Security v.s. Performance

MQDSS depends on four parameters q, k, n, r. The first parameter q determines the underlying field, and changing it has mostly to do with performance, since all the arithmetic operations are performed using different types of instructions which may influence speed for example. In some cases, the choice of q may introduce different dedicated attacks for the particular field, as in the case of $q = 2$, which may have slightly better performance (see Chapter 2 for detailed analysis of the known algorithms against the $\mathcal{MQ}$ problem). For a fixed value of q by increasing or decreasing the parameters k and n we increase or decrease the resistance of the system against known attacks. Note that we have specified earlier $r = \left\lceil k / \log_2 \frac{2q}{q+1} \right\rceil$ but it is possible (if one wants) to independently tune this parameter (for example to increase the performance). We will not consider this possibility in this document, and assume that r is not an independent parameter.

Based on the NIST call document [49], in a similar fashion to the 6 provided security categories, we identify 4 down-scaled categories

- BLOCKCIPHER64 (Category 0.1) - the security level of a generic block cipher with 64 bit key.
- HASHFUNCTION128 (Category 0.2) - the security level of a generic hash function with 128 bit output.
- BLOCKCIPHER96 (Category 0.3) - the security level of a generic block cipher with 96 bit key.
- HASHFUNCTION192 (Category 0.4) - the security level of a generic hash function with 192 bit output.

Our estimate of the concrete security level these provide in terms of classical and quantum gates, assuming black box treatment of the primitives (i.e. the best attacks are the generic ones) is given in Table 13.1.

	Security category	Classical Gates	Quantum Gates	Quantum circuit depth
0.1	BLOCKCIPHER64	2^{74}	2^{50}	2^{46}
0.2	HASHFUNCTION128	2^{77}		
0.3	BLOCKCIPHER96	2^{108}	2^{66}	2^{62}
0.4	HASHFUNCTION192	2^{110}		

Table 13.1: Basic characteristics of the scaled down parameter sets

We identify the following parameter sets that satisfy the scaled down security categories 0.1-0.4.

Security category	k	q	n	r	Public key size (bytes)	Secret key size (bytes)	Signature size (bytes)
0.1-0.2	64	4	48	95	20	8	6492
0.1-0.2	64	16	32	71	24	8	5712
0.1-0.2	64	31	24	68	23	8	5268
0.1-0.2	64	32	24	67	23	8	5191
0.1-0.2	64	64	24	66	26	8	5708
0.3-0.4	96	4	64	142	28	12	13680
0.3-0.4	96	16	40	106	32	12	11496
0.3-0.4	96	31	40	101	37	12	12471
0.3-0.4	96	32	40	101	37	12	12471
0.3-0.4	96	64	32	99	36	12	11928

Table 13.2: Basic characteristics of the scaled down parameter sets

Security category			Best classical attack		Best quantum attack		
	q	n	algorithm	Gates	algorithm	Gates	Depth
0.1-0.2	4	48	Crossbread	2^{84}	Crossbread	2^{57}	2^{48}
0.1-0.2	16	32	Crossbread	2^{87}	Crossbread	2^{59}	2^{51}
0.1-0.2	31	24	Crossbread	2^{83}	Crossbread	2^{59}	2^{50}
0.1-0.2	32	24	Crossbread	2^{84}	Crossbread	2^{53}	2^{45}
0.1-0.2	64	24	HybridF5	2^{94}	Crossbread	2^{60}	2^{52}
0.3-0.4	4	64	Crossbread	2^{109}	Crossbread	2^{71}	2^{62}
0.3-0.4	16	40	Crossbread	2^{112}	Crossbread	2^{72}	2^{63}
0.3-0.4	31	40	Crossbread	2^{135}	Crossbread	2^{86}	2^{76}
0.3-0.4	32	40	Crossbread	2^{136}	Crossbread	2^{83}	2^{73}
0.3-0.4	64	32	HybridF5	2^{120}	Crossbread	2^{73}	2^{64}

Table 13.3: Best classical and quantum attacks against the scaled down parameter sets

Since the signature size of MQDSS is the most critical performance characteristic, it is natural to consider it over other characteristics when estimating the security vs. performance trade-off. For better visual judgment, we have plotted this trade-off for $q = 31$, which is the chosen value of our reference parameter sets (see Section 8.1).

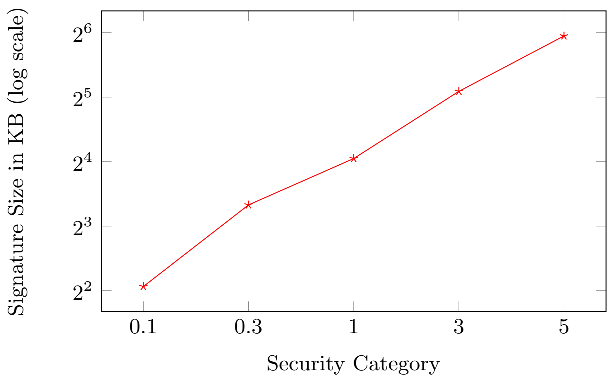

Fig. 13.1: Security category v.s. signature size

14

Strengths and Weaknesses

For any cryptographic design, the final product is a result based on decisions made to satisfy a certain security level, while maintaining desired properties such as performance and usability. This trade-off necessarily introduces weaknesses, but the designers' goal is to preserve enough good features to make the schemes attractive.

MQDSS is not an exception. In this chapter, we summarize and discuss the strengths and weaknesses of our proposal.

Strengths of MQDSS:

- *Small keys.* MQDSS has extremely small keys, comparable to contemporary schemes such as ECDSA that provide only classical security. On the other hand, they are several orders of magnitudes smaller than the keys of other $\mathcal{MQ}$ schemes.
- *Provably secure $\mathcal{MQ}$ signature, with reduction from the $\mathcal{MQ}$ problem.* MQDSS is the first multivariate signature scheme that is provably secure, and whose security relies solely on the $\mathcal{MQ}$ problem. The lack of security proofs throughout the history of $\mathcal{MQ}$ cryptography has made its representatives extremely prone to cryptanalysis and unfortunately the whole area has obtained bad reputation because of this. We believe that MQDSS together with the SSH schemes [43] are a step towards regaining confidence in $\mathcal{MQ}$ cryptography.
- *Flexible parameters.* All four parameters q, n, k, r can be easily tuned to match different security levels and platforms. Even more the number of variables is independent of the number of rounds, so in case of improvement in algebraic attacks against the $\mathcal{MQ}$ problem only the number of variable could be changed.
- *Simple design.* The underlying IDS uses a simple splitting technique based on the bilinearity of the polar form. The rest is a slightly more general Fiat-Shamir transform to turn the interactive protocol into a signature. The design does not utilize complicated algebraic structures (possibly even mathematically poorly understood), there is no dependence on possibly vulnerable distribution samplers, and in general there is very little room for flawed deployment.
- *Suitable for hardware implementation.* Due to the flexible parameters, it is possible to define MQDSS over fields of characteristic 2, such as $\mathbb{F}_{16}$ that are especially suitable for hardware implementation.
- *Naturally parallelizable.* The computations within a round are independent of the other rounds so it is straightforward to perform in parallel all rounds.

- *Inherently constant-time.* The straight-forward way of implementing the scheme is inherently protected against timing attacks. Evaluating the $\mathcal{MQ}$ function can traditionally be done in ways that depend on the input, but this is typically an additional optimization effort. Moreover, our chosen parameter set makes this unattractive on most platforms.

Weaknesses of MQDSS:

- *Large signature size.* Probably the biggest weakness of MQDSS is its signature size. Compared to traditional signature schemes the signature is at least 100 times larger. The same is true for other multivariate schemes. However, traditional $\mathcal{MQ}$ schemes have ad-hoc designs, without proof of security. Even more, in a typical usage scenario of signatures such as PKI, what matters is actually the size of the public key plus the signature. In such a setting MQDSS is still better, beating traditional $\mathcal{MQ}$ schemes by a factor of 2-20 depending on the scheme. On the other hand, provably secure schemes that provide post-quantum security tend to have much larger signatures, and for the schemes we are aware, the signatures are in the same range as MQDSS.
- *Security proof in the ROM, and not in the QROM.* In Section A.1 we showed in the random oracle model that $q2$-signature schemes are EU-CMA secure when the underlying IDS satisfies certain conditions. However, similar to the standard Fiat-Shamir transform, our proof relies on a forking lemma, which introduces two serious problems in the post-quantum setting (i.e. in the quantum accessible random oracle model): rewinding of the adversary, and adaptively programming the random oracle. While it is known how to deal with the latter [57], the former seems to be a serious obstacle [2]. The only known way to fix the Fiat-Shamir transform in the QROM setting [23] is using oblivious commitments, which are a certain kind of trapdoor commitments, effectively avoiding rewinding at the cost of introducing the necessity of a trapdoor function. This makes the solution not applicable in our setting as there are no known trapdoor functions with a reduction from the $\mathcal{MQ}$-problem.
  It is however possible to use a different transform that overcomes the problems of the forking lemma in the QROM. In [17], the authors generalize the Unruh transform [57], and apply it to the 5-pass SSH of Sakumoto *et al.*. The obtained signature scheme is secure in the ROM, but at a huge cost - the signature size becomes $\approx 120KB$ which in our opinion is not in the range of desired practicality.
- *Security proof not tight.* Another weakness of our security proof is that is not at all tight. This is again an inherent weakness introduced by the rewinding technique of the forking lemma. Therefore, in order to produce a tight security reduction for MQDSS one would have to base the proof on different techniques. At the moment, we are not aware of such techniques that we could use.

15

AVX2 Implementation of MQDSS

To demonstrate performance, we have also implemented the scheme using AVX2 vector instructions. As mentioned above, this makes convenient use of the structure of the terms, allowing implementations to benefit from the `vpmaddubs` instruction to combine two multiplications with an addition. In one instruction, this computes two 8 bit SIMD multiplications and a 16 bit SIMD addition. This also underlines the benefit of details such as elements in $\mathbf{F}$ in signed representation, since this allows accumulating more additions in vectorized 16-bit words before performing a reduction.

When arranging reductions, we must strike a careful balance between preventing overflow and not reducing more often than necessary. As we make extensive use of `vpmaddubsw`, which takes both a signed and an unsigned operand to compute the quadratic monomials, we ensure that the input variables for the $\mathcal{MQ}$ function are unsigned values (in particular: $\{0, \ldots, 31\}$). For the coefficients in the system parameter $\mathbf{F}$, we can then freely assume the values are in $\{-15, \ldots, 15\}$, as these are the direct result of a pseudo-random generator.

It turns out to be efficient to immediately reduce the quadratic monomials back to $\{0, \ldots, 31\}$ when they are computed. When we now multiply such a product with an element from the system parameter and add it to the accumulators, the maximum value of each accumulator word will be at most[1] $64 \cdot 31 \cdot 15 = 29760$. As this does not exceed the maximum value of 32768, we only have to perform reductions on each individual accumulator at the very end.

For the smaller parameter set, i.e. $n = 48$, these constraints are less pressing, but the maximum value accumulators remains in the same ballpark. Both $n = 48$ and $n = 64$ benefit from the fact that these parameters are multiples of 16, which results in a very similar optimal implementation strategy and convenient code reuse.

[1] This follows from the fact that we combine 64 such monomials in two `YMM` registers.

References

1. Abdalla, M., An, J.H., Bellare, M., Namprempre, C.: From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In: Knudsen, L.R. (ed.) Advances in Cryptology — EUROCRYPT 2002: International Conference on the Theory and Applications of Cryptographic Techniques Amsterdam, The Netherlands, April 28 – May 2, 2002 Proceedings. pp. 418–433. Springer Berlin Heidelberg, Berlin, Heidelberg (2002), https://doi.org/10.1007/3-540-46035-7_28
2. Ambainis, A., Rosmanis, A., Unruh, D.: Quantum attacks on classical proof systems: The hardness of quantum rewinding. In: FOCS 2014. pp. 474–483 (2014), http://eprint.iacr.org/2014/296
3. Amy, M., Maslov, D., Mosca, M.: Polynomial-time t-depth optimization of clifford+t circuits via matroid partitioning. IEEE Trans. on CAD of Integrated Circuits and Systems 33(10), 1476–1489 (2014), https://doi.org/10.1109/TCAD.2014.2341953
4. Amy, M., Maslov, D., Mosca, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Trans. on CAD of Integrated Circuits and Systems 32(6), 818–830 (2013), https://doi.org/10.1109/TCAD.2013.2244643
5. Bardet, M., Faugère, J., Salvy, B.: On the complexity of the F5 Gröbner basis algorithm. Journal of Symbolic Computation 70, 49–70 (2015), https://hal.inria.fr/hal-01064519/document
6. Bardet, M., Faugère, J.C., Salvy, B.: On the complexity of the F5 Gröbner basis algorithm. Journal of Symbolic Computation 70(Supplement C), 49 – 70 (2015), http://www.sciencedirect.com/science/article/pii/S0747717114000935
7. Bardet, M., Faugère, J., Salvy, B., Spaenlehauer, P.: On the complexity of solving quadratic boolean systems. Journal of Complexity 29(1), 53–75 (2013), www-polsys.lip6.fr/~jcf/Papers/BFSS12.pdf
8. Bardet, M., Faugère, J.C., Salvy, B., Yang, B.Y.: Asymptotic Behaviour of the Degree of Regularity of Semi-Regular Polynomial Systems. In: Proc. of MEGA 2005, Eighth International Symposium on Effective Methods in Algebraic Geometry (2005)
9. Beauregard, S., Brassard, G., Fernandez, J.M.: Quantum arithmetic on galois fields (2003)
10. Bernstein, D.J.: Multi-user schnorr security, revisited. Cryptology ePrint Archive, Report 2015/996 (2015), https://eprint.iacr.org/2015/996
11. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak reference (2011), http://keccak.noekeon.org/
12. Bettale, L., Faugère, J., Perret, L.: Solving polynomial systems over finite fields: improved analysis of the hybrid approach. In: van der Hoeven, J., van Hoeij, M. (eds.) Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation – ISSAC '12. pp. 67–74. ACM (2012), https://hal.inria.fr/hal-00776070/document
13. Bouillaguet, C., Chen, H.C., Cheng, C.M., Chou, T., Niederhagen, R., Shamir, A., Yang, B.Y.: Fast exhaustive search for polynomial systems in $\mathbb{F}_2$. In: Mangard, S., Standaert, F.X. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2010. LNCS, vol. 6225, pp. 203–218. Springer (2010), http://dx.doi.org/10.1007/978-3-642-15031-9_14, https://eprint.iacr.org/2010/313
14. Boyer, M., Brassard, G., Høyer, P., Tapp, A.: Tight bounds on quantum searching. Fortschritte der Physik 46(4-5), 493–505 (1998), http://dx.doi.org/10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P
15. Buchberger, B.: Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. J. Symb. Comput. 41(3-4), 475–511 (2006)

16. Chen, M.S., Hülsing, A., Rijneveld, J., Samardjiska, S., Schwabe, P.: From 5-pass $\mathcal{MQ}$-based identification to $\mathcal{MQ}$-based signatures. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016. LNCS, vol. 10032, pp. 135–165. Springer (2016), http://eprint.iacr.org/2016/708
17. Chen, M.S., Hülsing, A., Rijneveld, J., Samardjiska, S., Schwabe, P.: Sofia: Mq-based signatures in the qrom. Cryptology ePrint Archive, Report 2017/680 (2017), https://eprint.iacr.org/2017/680
18. Cheung, D., Maslov, D., Mathew, J., Pradhan, D.K.: On the design and optimization of a quantum polynomial-time attack on elliptic curve cryptography. In: Kawano, Y., Mosca, M. (eds.) Theory of Quantum Computation, Communication, and Cryptography: Third Workshop, TQC 2008 Tokyo, Japan, January 30 - February 1, 2008. Revised Selected Papers. pp. 96–104. Springer Berlin Heidelberg, Berlin, Heidelberg (2008), https://doi.org/10.1007/978-3-540-89304-2_9
19. Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing. pp. 1–6. STOC '87, ACM, New York, NY, USA (1987), http://doi.acm.org/10.1145/28395.28396
20. Coppersmith, D.: Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. Mathematics of Computation 62, 333–350 (1994)
21. Courtois, N., Klimov, E., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) Advances in Cryptology – EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer (2000), www.iacr.org/archive/eurocrypt2000/1807/18070398-new.pdf
22. Courtois, N.T.: Efficient zero-knowledge authentication based on a linear algebra problem MinRank. In: Boyd, C. (ed.) Advances in Cryptology – ASIACRYPT 2001. LNCS, vol. 2248, pp. 402–421. Springer (2001), https://eprint.iacr.org/2001/058
23. Dagdelen, Ö., Fischlin, M., Gagliardoni, T.: The Fiat–Shamir transformation in a quantum world. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology - ASIACRYPT 2013. LNCS, vol. 8270, pp. 62–81. Springer (2013), http://dx.doi.org/10.1007/978-3-642-42045-0_4, https://eprint.iacr.org/2013/245
24. Diem, C.: The XL-algorithm and a conjecture from commutative algebra. In: Lee, P.J. (ed.) Advances in Cryptology – ASIACRYPT 2004. LNCS, vol. 3329, pp. 323–337. Springer (2004), https://www.iacr.org/archive/asiacrypt2004/33290320/33290320.pdf
25. Ding, J., Hu, L., Yang, B.Y., Chen, J.M.: Note on design criteria for rainbow-type multivariates. Cryptology ePrint Archive, Report 2006/307 (2006), https://eprint.iacr.org/2006/307
26. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Security of the fiat-shamir transformation in the quantum random-oracle model. Cryptology ePrint Archive, Report 2019/190 (2019), https://eprint.iacr.org/2019/190
27. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra 139, 61–88 (1999), http://www-polsys.lip6.fr/~jcf/Papers/F99a.pdf
28. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In: Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation – ISSAC '02. pp. 75–83. ACM (2002), http://www-polsys.lip6.fr/~jcf/Papers/F02a.pdf
29. Faugère, J.C., Levy-dit-Vehel, F., Perret, L.: Cryptanalysis of MinRank. In: Wagner, D. (ed.) Advances in Cryptology – CRYPTO 2008. LNCS, vol. 5157, pp. 280–296. Springer (2008), http://www-polsys.lip6.fr/~jcf/Papers/crypto08.pdf
30. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology — CRYPTO' 86: Proceedings. pp. 186–194. Springer Berlin Heidelberg, Berlin, Heidelberg (1987), https://doi.org/10.1007/3-540-47721-7_12
31. Fusco, G., Bach, E.: Phase transition of multivariate polynomial systems. In: Cai, J.Y., Cooper, B.S., Zhu, H. (eds.) International Conference on Theory and Applications of Models of Computation – TAMC 2007. LNCS, vol. 4484, pp. 632–645. Springer (2007), https://minds.wisconsin.edu/bitstream/handle/1793/60544/TR1588.pdf
32. Galbraith, S., Malone-Lee, J., Smart, N.P.: Public key signatures in the multi-user setting. Inf. Process. Lett. 83(5), 263–266 (Sep 2002), http://dx.doi.org/10.1016/S0020-0190(01)00338-6
33. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company (1979)
34. Giesbrecht, M., Lobo, A., Saunders, B.D.: Certifying inconsistency of sparse linear systems. In: Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation – ISSAC '98. pp. 113–119 (1998), http://doi.acm.org/10.1145/281508.281591, https://cs.uwaterloo.ca/~mwg/files/incons.pdf

35. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (1988), https://people.csail.mit.edu/silvio/Selected%20Scientific%20Papers/Digital%20Signatures/A_Digital_Signature_Scheme_Secure_Against_Adaptive_Chosen-Message_Attack.pdf
36. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying grover's algorithm to aes: Quantum resource estimates. In: Takagi, T. (ed.) Post-Quantum Cryptography: 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings. pp. 29–43. Springer International Publishing, Cham (2016), https://doi.org/10.1007/978-3-319-29360-8_3
37. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing – STOC '96. pp. 212–219. ACM (1996), https://arxiv.org/pdf/quant-ph/9605043v3.pdf
38. Joux, A., Vitse, V.: A crossbred algorithm for solving boolean polynomial systems. Cryptology ePrint Archive, Report 2017/372 (2017), http://eprint.iacr.org/2017/372
39. Kepley, S., Steinwandt, R.: Quantum circuits for $\mho_{2}$n-multiplication with subquadratic gate count. Quantum Information Processing 14(7), 2373–2386 (2015), https://doi.org/10.1007/s11128-015-0993-1
40. Kiltz, E., Lyubashevsky, V., Schaffner, C.: A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018. pp. 552–586. Springer International Publishing, Cham (2018)
41. Kiltz, E., Masny, D., Pan, J.: Optimal security proofs for signatures from identification schemes. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II. pp. 33–61. Springer Berlin Heidelberg, Berlin, Heidelberg (2016), https://doi.org/10.1007/978-3-662-53008-5_2
42. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar signature schemes. In: Stern, J. (ed.) Advances in Cryptology – EUROCRYPT '99. LNCS, vol. 1592, pp. 206–222. Springer (1999), http://www.goubin.fr/papers/OILLONG.PDF
43. KoichiSakumoto, Shirai, T., Hiwatari, H.: Public-key identification schemes based on multivariate quadratic polynomials. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011. LNCS, vol. 6841, pp. 706–723. Springer (2011), https://www.iacr.org/archive/crypto2011/68410703/68410703.pdf
44. Lazard, D.: Gröbner-Bases, Gaussian elimination and resolution of systems of algebraic equations. In: van Hulzen, J.A. (ed.) EUROCAL. Lecture Notes in Computer Science, vol. 162, pp. 146–156. Springer (1983)
45. Leichtle, D.: Post-quantum signatures from identification schemes. Master Thesis, Technicshe Universiteit Eindhoven (2018)
46. Lokshtanov, D., Paturi, R., Tamaki, S., Williams, R.R., Yu, H.: Beating brute force for systems of polynomial equations over finite fields. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19. pp. 2190–2202 (2017), https://doi.org/10.1137/1.9781611974782.143
47. Montgomery, P.L.: A block lanczos algorithm for finding dependencies over gf(2). In: Guillou, L.C., Quisquater, J.J. (eds.) Advances in Cryptology — EUROCRYPT '95: International Conference on the Theory and Application of Cryptographic Techniques Saint-Malo, France, May 21–25, 1995 Proceedings. pp. 106–120. Springer Berlin Heidelberg, Berlin, Heidelberg (1995), https://doi.org/10.1007/3-540-49264-X_9
48. NIST: FIPS PUB 202 – SHA-3 standard: Permutation-based hash and extendable-output functions (2015), http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
49. NIST: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. Cryptology ePrint Archive, Report 2015/996 (2016), http://csrc.nist.gov/groups/ST/postquantum-crypto/documents/call-for-proposals-final-dec-2016.pdf
50. Ohta, K., Okamoto, T.: On concrete security treatment of signatures derived from identification. In: Krawczyk, H. (ed.) Advances in Cryptology — CRYPTO '98: 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings. pp. 354–369. Springer Berlin Heidelberg, Berlin, Heidelberg (1998), https://doi.org/10.1007/BFb0055741
51. Patarin, J.: Hidden field equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In: Maurer, U. (ed.) Advances in Cryptology – EUROCRYPT '96. LNCS, vol. 1070, pp. 33–48. Springer (1996), http://www.minrank.org/hfe.pdf

52. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U. (ed.) Advances in Cryptology – EUROCRYPT '96, LNCS, vol. 1070, pp. 387–398. Springer (1996), https://www.di.ens.fr/~pointche/Documents/Papers/1996_eurocrypt.pdf
53. Stern, J.: A new paradigm for public key identification. IEEE Transactions on Information Theory 42(6), 1757–1768 (1996), https://www.di.ens.fr/users/stern/data/St55b.pdf
54. Strassen, V.: Gaussian elimination is not optimal. Numer. Math. 13(4), 354–356 (Aug 1969), http://dx.doi.org/10.1007/BF02165411
55. Thomae, E.: About the Security of Multivariate Quadratic Public Key Schemes. Ph.D. thesis, Ruhr-University Bochum, Germany (2013), https://www.iacr.org/phds/116_EnricoThomae_AboutSecurityMultivariateQuadr.pdf
56. Thomae, E., Wolf, C.: Solving underdetermined systems of multivariate quadratic equations revisited. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) Public Key Cryptography – PKC 2012: 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings. pp. 156–171. Springer Berlin Heidelberg, Berlin, Heidelberg (2012), https://doi.org/10.1007/978-3-642-30057-8_10
57. Unruh, D.: Non-interactive zero-knowledge proofs in the quantum random oracle model. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. LNCS, vol. 9056, pp. 755–784. Springer (2015), http://dx.doi.org/10.1007/978-3-662-46803-6_25, http://eprint.iacr.org/2014/587
58. Unruh, D.: Post-quantum security of fiat-shamir. Cryptology ePrint Archive, Report 2017/398, to appear in Advances in Cryptology - ASIACRYPT 2017 (2017), https://eprint.iacr.org/2017/398
59. Westerbaan, B., Schwabe, P.: Solving binary $\mathcal{MQ}$ with grover's algorithm. In: Carlet, C., Hasan, A., Saraswat, V. (eds.) Security, Privacy, and Advanced Cryptography Engineering. LNCS, vol. 10076. Springer (2016), document ID: 40eb0e1841618b99ae343ffa073d6c1e, http://cryptojedi.org/papers/#mqgrover
60. Williams, V.V.: Multiplying matrices faster than coppersmith-winograd. In: Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing. pp. 887–898. STOC '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2213977.2214056
61. Yang, B., Chen, J.: Theoretical analysis of XL over small fields. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) Information Security and Privacy. LNCS, vol. 3108, pp. 277–288. Springer (2004), http://dx.doi.org/10.1007/978-3-540-27800-9_24, http://www.iis.sinica.edu.tw/papers/byyang/2386-F.pdf
62. Yang, B.Y., Chen, J.M.: All in the XL family: Theory and practice. In: sik Park, C., Chee, S. (eds.) Information Security and Cryptology – ICISC 2004. pp. 67–86. Springer (2005), http://by.iis.sinica.edu.tw/by-publ/recent/xxl.pdf
63. Yeh, J.Y.C., Cheng, C.M., Yang, B.Y.: Operating Degrees for XL vs. F4/F5 for Generic $\mathcal{MQ}$ with Number of Equations Linear in That of Variables. In: Fischlin, M., Katzenbeisser, S. (eds.) Number Theory and Cryptography: Papers in Honor of Johannes Buchmann on the Occasion of His 60th Birthday. pp. 19–33. Springer (2013), http://www.iis.sinica.edu.tw/papers/byyang/17377-F.pdf

A

Security proofs

A.1 Security of $q2$-signature schemes.

For completeness of this document, we provide in full the security reduction for Construction 5.1 in the random oracle model, and the proof of EU-CMA security of the obtained signature scheme.

To prove this claim, we proceed in several steps. The proof builds on techniques introduced by Pointcheval and Stern [52] (see Section 4.2 for a brief description of the technique). As the reduction is far from being tight, we refrain from doing an exact proof as it does not gain us anything but a complicated statement. We first recall an important tool from [52] called the splitting lemma.

Lemma A.1 (Splitting lemma [52]). *Let $A \subset X \times Y$, such that* $\Pr[A(x, y)] \geqslant \epsilon$. *Then, there exists $\Omega \subset X$, such that*

$$\Pr[x \in \Omega] \geqslant \epsilon/2,$$
$$\Pr[A(a, y) | a \in \Omega] \geqslant \epsilon/2.$$

We next present a forking lemma for $q2$-signature schemes. The lemma shows that we can obtain four valid signatures which contain four valid transcripts of the underlying IDS, given a successful key-only adversary. Moreover, these four transcripts fulfill a certain requirement on the challenges (here the related parts of the hash function outputs) that we need later.

In the lemma and in the rest of the chapter, we model the functions H_1 and H_2 as independent random oracles $\mathcal{O}_1$ and $\mathcal{O}_2$. Furthermore, for ease of exposition in our proofs, we use a "full" version of a signature, including the outputs h_1 and h_2 of H_1 and H_2, i.e., instead of $\sigma = (\sigma_0, \sigma_1, \sigma_2)$, we assume a signature has the form $\sigma = (\sigma_0, \mathsf{h}_1, \sigma_1, \mathsf{h}_2, \sigma_2)$. (Note that h_1 and h_2 need not be included in the signatures because they can be easily reconstructed from the other values.)

Lemma A.2 (Forking lemma for $q2$-signature schemes). *Let $\mathsf{Dss}(1^k)$ be a $q2$-signature scheme with security parameter $k \in \mathbb{N}$. If there exists a PPT adversary $\mathcal{A}$ that can output a valid signature message pair (M, σ) with non-negligible success probability, given only the public key as input, then, with non-negligible probability, rewinding $\mathcal{A}$ a polynomial number of times with the same random tape but different oracles, outputs 4 valid message signature pairs $(M, \sigma = (\sigma_0, \mathsf{h}_1, \sigma_1, \mathsf{h}_2, \sigma_2))$, $(M, \sigma' = (\sigma_0, \mathsf{h}_1', \sigma_1', \mathsf{h}_2', \sigma_2'))$,*

$(M, \sigma'' = (\sigma_0, \mathsf{h}''_1, \sigma''_1, \mathsf{h}''_2, \sigma''_2))$, $(M, \sigma''' = (\sigma_0, \mathsf{h}'''_1, \sigma'''_1, \mathsf{h}'''_2, \sigma'''_2))$, such that there exists $j \in \{1, \ldots, r\}$ such that:

$$\begin{aligned}(\mathsf{ch}_1)^{(j)} = (\mathsf{ch}'_1)^{(j)} \neq (\mathsf{ch}''_1)^{(j)} = (\mathsf{ch}'''_1)^{(j)},\\ (\mathsf{ch}_2)^{(j)} = (\mathsf{ch}''_2)^{(j)} \neq (\mathsf{ch}'_2)^{(j)} = (\mathsf{ch}'''_2)^{(j)},\end{aligned} \tag{A.1}$$

where $\mathsf{h}_1 = (\mathsf{ch}_1^{(1)}, \mathsf{ch}_1^{(2)}, \ldots, \mathsf{ch}_1^{(r)})$ and $\mathsf{h}_2 = (\mathsf{ch}_2^{(1)}, \mathsf{ch}_2^{(2)}, \ldots, \mathsf{ch}_2^{(r)})$ and similarly for $\mathsf{h}'_1, \mathsf{h}''_1, \mathsf{h}'''_1$ and $\mathsf{h}'_2, \mathsf{h}''_2, \mathsf{h}'''_2$.

Proof. To prove the Lemma we need to show that we can rewind $\mathcal{A}$ three times (and adaptability program the random oracles) and at the same time, the probability that $\mathcal{A}$ succeeds in forging a (different) signature in all four runs is non-negligible. Moreover, we have to show that the signatures have the additional property claimed in the Lemma, again with non-negligible probability.

Let $\omega \in R_w$ be $\mathcal{A}$'s random tape with R_w the set of allowable random tapes. During the attack $\mathcal{A}$ may ask polynomially many queries (in the security parameter k) $Q_1(k)$ and $Q_2(k)$ to the random oracles $\mathcal{O}_1$ and $\mathcal{O}_2$. Let $q_{1,1}$, $q_{1,2}, \ldots,$ q_{1,Q_1} and $q_{2,1}$, $q_{2,2}, \ldots,$ q_{2,Q_2} be the queries to $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively. Moreover, let $(r_{1,1}, r_{1,2}, \ldots, r_{1,Q_1}) \in (\mathsf{C}_1^r)^{Q_1}$ and $(r_{2,1}, r_{2,2}, \ldots, r_{2,Q_2}) \in (\mathsf{C}_2^r)^{Q_2}$ the corresponding answers of the oracles.

Denote by F the event that $\mathcal{A}$ outputs a valid message signature pair $(M, \sigma = (\sigma_0, \mathsf{h}_1, \sigma_1, \mathsf{h}_2, \sigma_2))$. Per assumption, this event occurs with non-negligible probability, i.e., $\Pr[\mathsf{F}] = \frac{1}{P(k)}$, for some polynomial $P(k)$. In addition, F implies $\mathsf{h}_1 = \mathcal{O}_1(M, \sigma_0)$ and $\mathsf{h}_2 = \mathcal{O}_2(M, \sigma_0, \mathsf{h}_1, \sigma_1)$. As $\mathsf{h}_1, \mathsf{h}_2$ are chosen uniformly at random from exponentially large sets $\mathsf{C}_1^r, \mathsf{C}_2^r$, the probability that $\mathcal{A}$ did not query $\mathcal{O}_1$ with (M, σ_0) and $\mathcal{O}_2$ with $(M, \sigma_0, \mathsf{h}_1, \sigma_1)$ is negligible. Hence, there exists a polynomial P' such that the event F' that F occurs and $\mathcal{A}$ queried $\mathcal{O}_1$ with (M, σ_0) and $\mathcal{O}_2$ with $(M, \sigma_0, \mathsf{h}_1, \sigma_1)$ has probability $\Pr[\mathsf{F}'] = \dfrac{1}{P'(k)}$.

For the moment consider only the second oracle. From the previous equation, there exists at least one $\beta \leqslant Q_2$ such that

$$\Pr[\mathsf{F}' \wedge q_{2,\beta} = (M, \sigma_0, \mathsf{h}_1, \sigma_1)] \geqslant \frac{1}{Q_2(k)P'(k)}$$

where the probability is taken over the random coins of $\mathcal{A}$ and all answers from $\mathcal{O}_2$, i.e. over the set $\mathcal{B} = \{(\omega, r_{2,1}, r_{2,2}, \ldots, r_{2,Q_2}) | \omega \in R_w \wedge (r_{2,1}, r_{2,2}, \ldots, r_{2,Q_2}) \in (\mathsf{C}_2^r)^{Q_2} \wedge \mathsf{F}' \wedge q_{2,\beta} = (M, \sigma_0, \mathsf{h}_1, \sigma_1)\}$.

(Informally, the following steps just show that the success of an algorithm with non-negligible success probability cannot be conditioned on an event that occurs only with negligible probability (i.e. the outcome of the $q_{2,\beta}$ query landing in some negligible subset).)

The last equation implies that there exists a non-negligible set of "good" random tapes $\Omega_\beta \subseteq R_\omega$ for which $\mathcal{A}$ can provide a valid signature and $q_{2,\beta}$ is the oracle query determining h_2. Applying the splitting lemma, we get that

$$\begin{aligned}\Pr[w \in \Omega_\beta] &\geqslant \frac{1}{2Q_2(k)P'(k)}\\ \Pr[(\omega, r_{2,1}, r_{2,2}, \ldots, r_{2,Q_2}) \in \mathcal{B} | w \in \Omega_\beta] &\geqslant \frac{1}{2Q_2(k)P'(k)}\end{aligned}$$

Applying the same reasoning again we can derive from the later probability being non-negligible that there exists a non-negligible subset $\Omega_{\beta,\omega}$ of the "good" oracle responses

$(r_{2,1}, r_{2,2}, \dots, r_{2,\beta-1})$ such that $(\omega, r_{2,1}, r_{2,2}, \dots, r_{2,Q_2}) \in \mathcal{B}$. Applying the splitting lemma again, we get

$$\Pr[(r_{2,1}, \dots, r_{2,\beta-1}) \in \Omega_{\beta,\omega}] \geqslant \frac{1}{4Q_2(k)P'(k)}$$
$$\Pr[(\omega, r_{2,1}, \dots, r_{2,Q_2}) \in \mathcal{B} | (r_{2,1}, \dots, r_{2,\beta-1}) \in \Omega_{\beta,\omega})] \geqslant \frac{1}{4Q_2(k)P'(k)}$$

This means that rewinding $\mathcal{A}$ to the point where it made query $q_{2,\beta}$ and running it with new, random $r'_{2,\beta}, \dots, r'_{2,Q_2}$ has a non-negligible probability of $\mathcal{A}$ outputting another valid signature. Therefore, we can use $\mathcal{A}$ to find with non-negligible probability two valid message signature pairs $(M, \sigma = (\sigma_0, \mathsf{h}_1, \sigma_1, \mathsf{h}_2, \sigma_2))$, $(M, \sigma' = (\sigma_0, \mathsf{h}'_1, \sigma'_1, \mathsf{h}'_2, \sigma'_2))$, such that $(\sigma_0, \mathsf{h}_1, \sigma_1) = (\sigma_0, \mathsf{h}'_1, \sigma'_1)$. and $\mathsf{h}_2 \neq \mathsf{h}'_2$.

We now rewind the adversary again using exactly the same technique as above but now considering the queries to $\mathcal{O}_1$ and its responses. In the replay we change the responses of $\mathcal{O}_1$ to obtain a third signature that differs from the previously obtained ones in the first associated hash value. In the same manner, it can be shown that with non-negligible probability $\mathcal{A}$ will output a third signature on M, $\sigma'' = (\sigma_0, \mathsf{h}''_1, \sigma''_1, \mathsf{h}''_2, \sigma''_2)$, such that $\mathsf{h}''_1 \neq \mathsf{h}'_1 = \mathsf{h}_1$.

Finally, we rewind the adversary a third time, keeping the responses of $\mathcal{O}_1$ from the last rewind and focusing on $\mathcal{O}_2$ again. Again, with non-negligible probability $\mathcal{A}$ will produce yet another signature on M, $\sigma''' = (\sigma_0, \mathsf{h}'''_1, \sigma'''_1, \mathsf{h}'''_2, \sigma'''_2)$ such that $(\sigma_0, \mathsf{h}''_1, \sigma''_1) = (\sigma_0, \mathsf{h}'''_1, \sigma'''_1)$ and $\mathsf{h}''_2 \neq \mathsf{h}'''_2$.

Summing up , rewinding the adversary three times, we can find four valid signatures $\sigma, \sigma', \sigma'', \sigma'''$ with non-negligible success probability $\frac{1}{P(k)}$ for some polynomial $P(k)$. Let us denote this event by $\mathcal{E}_\sigma$. So we have that

$$\Pr[\mathcal{E}_\sigma] \geqslant \frac{1}{P(k)}.$$

What remains is to show that the obtained signatures satisfy the particular structure from the lemma (Equation A.1) with non-negligible probability.

Let $\mathcal{H}$ be the event that for $(\sigma, \sigma', \sigma''.\sigma''')$ there exists a $j \in \{1, \dots, r\}$ such that (A.1) is satisfied. For the probability that $\mathcal{A}$ outputs a valid signature with this property, we have that

$$\mathsf{Pr}[\mathcal{E}_\sigma \wedge \mathcal{H}] = \mathsf{Pr}[\mathcal{E}_\sigma] - \mathsf{Pr}[\neg\mathcal{H} \wedge \mathcal{E}_\sigma] \geqslant \frac{1}{P(k)} - \mathsf{Pr}[\neg\mathcal{H} \wedge \mathcal{E}_\sigma]$$

Now, let $\sigma, \sigma', \sigma'', \sigma'''$ be the four valid signatures that $\mathcal{A}$ outputs under the event $\neg\mathcal{H} \wedge \mathcal{E}_\sigma$. This means that (A.1) is not satisfied for $\sigma, \sigma', \sigma'', \sigma'''$ for any $j \in \{1, \dots, r\}$.

Consider the set $S_{\neg\mathcal{H}}$ of all tuples $(\mathsf{h}_1, \mathsf{h}''_1, \mathsf{h}_2, \mathsf{h}'_2, \mathsf{h}''_2, \mathsf{h}'''_2) \in (\mathsf{C}_1^r)^2 \times (\mathsf{C}_2^r)^4$ where $\mathsf{h}_1 = (\mathsf{ch}_1^{(1)}, \mathsf{ch}_1^{(2)}, \dots, \mathsf{ch}_1^{(r)}) \in \mathsf{C}_1^r$, (similarly for h'_1), and $\mathsf{h}_2 = (\mathsf{ch}_2^{(1)}, \mathsf{ch}_2^{(2)}, \dots, \mathsf{ch}_2^{(r)}) \in \mathsf{C}_2^r$, (similarly for $\mathsf{h}'_2, \mathsf{h}''_2, \mathsf{h}'''_2$), and such that for every $j \in \{1, \dots, r\}$ at least one of the following is true:

$$i.(\mathsf{ch}_1)^{(j)} = (\mathsf{ch}'_1)^{(j)}; \quad ii.(\mathsf{ch}_2)^{(j)} = (\mathsf{ch}'_2)^{(j)}; \quad iii.(\mathsf{ch}''_2)^{(j)} = (\mathsf{ch}'''_2)^{(j)}.$$

It is clear that the hash value tuple $(\mathsf{h}_1, \mathsf{h}''_1, \mathsf{h}_2, \mathsf{h}'_2, \mathsf{h}''_2, \mathsf{h}'''_2)$ in $\mathcal{A}$'s output under the event $\neg\mathcal{H} \wedge \mathcal{E}_\sigma$ must be in $S_{\neg\mathcal{H}}$. Indeed if the hash value tuple does not come from $S_{\neg\mathcal{H}}$, then there exists a $j \in \{1, \dots, r\}$, such that none of $i., ii., iii.$, holds true, i.e., for this j

$$(\mathsf{ch}_1)^{(j)} \neq (\mathsf{ch}'_1)^{(j)} \wedge (\mathsf{ch}_2)^{(j)} \neq (\mathsf{ch}'_2)^{(j)} \wedge (\mathsf{ch}''_2)^{(j)} \neq (\mathsf{ch}'''_2)^{(j)}.$$

A little thought reveals that the last is equivalent to (A.1), which is a contradiction to the assumption that the tuple comes under the event $\neg\mathcal{H} \wedge \mathcal{E}_\sigma$.

Recall that for $q2$-signatures, C_1 has size q and C_2 size 2. Now, the cardinality of $S_{\neg\mathcal{H}}$ can be calculated to be $|S_{\neg\mathcal{H}}| = (4q(3q+1))^r$, whereas the cardinality of $(\mathsf{C}_1^r)^2 \times (\mathsf{C}_2^r)^4$ is $(16q^2)^r$. This means that

$$\mathsf{Pr}[\neg\mathcal{H} \wedge \mathcal{E}_\sigma] \leqslant \frac{(4q(3q+1))^r}{(16q^2)^r} = \left(\frac{3q+1}{4q}\right)^r,$$

which is negligible in k since according to Construction 5.1, the number of rounds r must be super-logarithmic (in k), to fulfill C_2^r being exponentially large (in k).

Finally,

$$\mathsf{Pr}[\mathcal{E}_\sigma \wedge \mathcal{H}] = \mathsf{Pr}[\mathcal{E}_\sigma] - \mathsf{Pr}[\neg\mathcal{H} \wedge \mathcal{E}_\sigma] \geqslant \frac{1}{P(k)} - \left(\frac{3q+1}{4q}\right)^r = \frac{1}{P(k)} - \mathrm{negl}(k)$$

and hence, the conditions from the lemma are satisfied with non-negligible probability. □

With Lemma A.2 we can already establish unforgeability under key only attacks:

Theorem A.3 (KOA security of $q2$-signature schemes). *Let $k \in \mathbb{N}$, $\mathsf{IDS}(1^k)$ a $q2$-IDS that has a key relation R, is KOW secure, and has a $q2$-extractor. Then $q2\text{-}\mathsf{Dss}(1^k)$, the $q2$-signature scheme derived applying Construction 5.1 is unforgeable under key-only attacks.*

Proof. Let $\mathcal{A}$ be a PPT algorithm that forges a signature in a KOA setting, i.e., given only the public key pk outputs a valid message-signature pair (M, σ) with non-negligible probability ϵ. We show how to construct an algorithm $\mathcal{M}^{\mathcal{A}}$ that given IDS public key and oracle access to $\mathcal{A}$ breaks the KOW security of IDS in essentially the same running time as the given $\mathcal{A}$ and with negligibly different success probability.

On input the IDS public key pk, $\mathcal{M}^{\mathcal{A}}$ runs $\mathcal{A}(\mathsf{pk})$ which outputs a valid message-signature pair (M, σ) for $q2\text{-}\mathsf{Dss}$. Using the technique from Lemma A.2, rewinding $\mathcal{A}$, $\mathcal{M}^{\mathcal{A}}$ obtains four valid signatures that with overwhelming probability contain four valid transcripts that satisfy Equation (A.1). These are exactly the type of transcripts needed for the $q2$-extractor to extract a valid secret key sk'. Since $(\mathsf{pk}, \mathsf{sk}') \in R$, $\mathcal{M}^{\mathcal{A}}$ breaks the KOW security of IDS. □

For $\mathsf{EU\text{-}CMA}$ security, we still have to deal with signature queries. The following lemma shows that a reduction can produce valid responses to the adversarial signature queries if the identification scheme is computationally honest-verifier zero-knowledge.

Lemma A.4. *Let $k \in \mathbb{N}$ the security parameter, $\mathsf{IDS}(1^k)$ a $q2$-IDS that is honest-verifier zero-knowledge. Then any PPT adversary $\mathcal{B}$ against the $\mathsf{EU\text{-}CMA}$-security of $q2\text{-}\mathsf{Dss}(1^k)$, the $q2$-signature scheme derived by applying Construction 5.1, can be turned into a key-only adversary $\mathcal{A}$ against $q2\text{-}\mathsf{Dss}$ with the properties described in Lemma A.2. $\mathcal{A}$ runs in polynomial time and succeeds with essentially the same success probability as $\mathcal{B}$.*

Proof. By construction. We show how to construct an oracle machine $\mathcal{A}^{\mathcal{B},\mathcal{S},\mathcal{O}_1,\mathcal{O}_2}$ that has access to $\mathcal{B}$, a computationally honest-verifier zero-knowledge simulator $\mathcal{S}$, and random oracles $\mathcal{O}_1, \mathcal{O}_2$. $\mathcal{A}$ produces a valid signature for $q2\text{-}\mathsf{Dss}(1^k)$ given only a public key running in time polynomial in k and achieving essentially the same success probability (up to a negligible difference) as $\mathcal{B}$.

Upon input of public key pk, $\mathcal{A}$ runs $\mathcal{B}^{\mathcal{O}'_1,\mathcal{O}'_2,\mathsf{Sign}}(\mathsf{pk})$ simulating the random oracles (ROs) $\mathcal{O}'_1, \mathcal{O}'_2$, as well as the signing oracle Sign towards $\mathcal{B}$. When $\mathcal{B}$ outputs a forgery (M^*, σ^*), $\mathcal{A}$ just forwards it.

To simulate the ROs, $\mathcal{A}$ keeps two initially empty tables of query-response pairs, one per oracle. Whenever $\mathcal{B}$ queries $\mathcal{O}'_b$, $\mathcal{A}$ first checks if the table for $\mathcal{O}'_b$ already contains a pair for this query. If such a pair exists, $\mathcal{A}$ just returns the stored response. Otherwise, $\mathcal{A}$ forwards the query to its own $\mathcal{O}_b$.

As IDS is computationally honest-verifier zero-knowledge there exists a PPT simulator $\mathcal{S}$ that upon input of a IDS public key generates a valid transcript which only negligibly reduced the success probability of an adversary compared to transcripts generated by honest protocol executions. Whenever $\mathcal{B}$ queries the signature oracle with message m, $\mathcal{A}$ runs $\mathcal{S}$ r times, to obtain r valid transcripts. $\mathcal{A}$ combines the transcripts to obtain a valid signature $\sigma = (\sigma_0, \mathsf{h}_1, \sigma_1, \mathsf{h}_2, \sigma_2)$. Before outputting σ, $\mathcal{A}$ checks if the table for $\mathcal{O}'_1$ already contains an entry for query (M, σ_0). If so, $\mathcal{A}$ aborts. Otherwise, $\mathcal{A}$ adds the pair $((M, \sigma_0), \mathsf{h}_1)$. Then, $\mathcal{A}$ checks the second table for query $(M, \sigma_0, \mathsf{h}_1, \sigma_1)$. Again, $\mathcal{A}$ aborts if it finds such an entry and adds $((M, \sigma_0, \mathsf{h}_1, \sigma_1), \mathsf{h}_2)$, otherwise.

The probability that $\mathcal{A}$ aborts is negligible in k. When answering signature queries, $\mathcal{A}$ verifies that certain queries were not made before. Both queries contain σ_1 which takes any given value only with negligible probability. On the other hand, the total number of queries that $\mathcal{B}$ makes to all its oracles is polynomially bounded. Hence, the probability that one of the two queries was already made before is negligible. If $\mathcal{A}$ does not abort, it perfectly simulates the random oracles towards $\mathcal{B}$. Because of the computationally HVZK, there is only a negligible difference in the success probability of $\mathcal{B}$ – and thereby $\mathcal{A}$ – compared to the real $\mathsf{EU\text{-}CMA}$ game in this case. Hence, $\mathcal{A}$ succeeds with essentially the same probability as $\mathcal{B}$. $\square$

We now got everything we need to prove EU-CMA security. The proof is a straight forward application of Lemma A.2 and Lemma A.4.

Theorem A.5 (EU-CMA security of $q2$-signature schemes). *Let $k \in \mathbb{N}$, $\mathsf{IDS}(1^k)$ a $q2$-IDS that has a key relation R, is KOW secure, is computationally honest-verifier zero-knowledge, and has a $q2$-extractor $\mathcal{E}$. Then $q2\text{-}\mathsf{Dss}(1^k)$, the $q2$-signature scheme derived applying Construction 5.1 is existentially unforgeable under adaptive chosen message attacks.*

Proof. Towards a contradiction, assume that there exists a PPT adversary $\mathcal{A}$ against the $\mathsf{EU\text{-}CMA}$-security of $q2\text{-}\mathsf{Dss}$ succeeding with non-negligible probability. We show how to construct a PPT algorithm $\mathcal{M}^{\mathcal{A}}$ that given the IDS public key and oracle access to $\mathcal{A}$ breaks the KOW security of IDS. Applying Lemma A.4, $\mathcal{M}^{\mathcal{A}}$ can construct a PPT key-only forger $\mathcal{B}$, with essentially the same success probability as $\mathcal{A}$. Given a public key for IDS (which is a valid $q2\text{-}\mathsf{Dss}$ public key) $\mathcal{M}^{\mathcal{A}}$ runs $\mathcal{B}$ as described in Lemma A.2. That way $\mathcal{M}^{\mathcal{A}}$ can use $\mathcal{B}$ to obtain four signatures that per (A.1) lead to four transcripts as required by the $q2$-extractor $\mathcal{E}$. Running $\mathcal{E}$, $\mathcal{M}^{\mathcal{A}}$ can extract a valid secret key sk' that breaks the KOW security of IDS.

$\mathcal{M}^{\mathcal{A}}$ just runs $\mathcal{B}$ and $\mathcal{E}$, two PPT algorithms. Consequently, $\mathcal{M}^{\mathcal{A}}$ runs in polynomial time. Also, $\mathcal{B}$ and $\mathcal{E}$ both have non-negligible success probability implying that $\mathcal{M}^{\mathcal{A}}$ also succeeds with non-negligible probability. □

A.2 Proof of Theorem 10.1 [EU-CMA security of MQDSS]

Before we present the proof, note that as our results from Section A.1 are non-tight we only prove an asymptotic statement. While this does not suffice to make any statement about the security of a specific parameter choice, it provides evidence that the general approach leads to a secure scheme.

To prove this theorem we would like to apply Theorem 5.2 (the same as Theorem A.5). However, Theorem 5.2 was formulated for a slightly more generic construction (see Construction 5.1). The point is that we apply an optimization originally proposed in [53]. So, in our actual proposal (see Chapter 7), the parallel composition of the IDS is slightly different as, instead of the commitments, only the hash of their concatenation is sent (c.f. σ_0 in Figure 7.2). Also, the last message (c.f. σ_2 in Figure 7.2) now contains the remaining commitments. Let's call this optimized version $opt\text{-}q2\text{-}\mathsf{Dss}(1^k)$.

Note that since MQDSS is an $opt\text{-}q2\text{-}\mathsf{Dss}(1^k)$ signature scheme, we could have focused our attention solely to $opt\text{-}q2\text{-}\mathsf{Dss}(1^k)$ schemes already in Chapter 5. However, this would have limited the general applicability of the result, as the above optimization is only applicable to schemes with a certain, less generic, structure such as MQDSS.

As the next Corollary shows, it is easy to verify that the results from Chapter 5 hold for the optimized $opt\text{-}q2\text{-}\mathsf{Dss}(1^k)$ scheme as well.

Corollary A.6 (EU-CMA security of $q2$-signature schemes). *Let $k \in \mathbb{N}$, $\mathsf{IDS}(1^k)$ a $q2$-IDS that has a key relation R, is KOW secure, is computationally honest-verifier zero-knowledge, and has a $q2$-extractor $\mathcal{E}$. Then $opt\text{-}q2\text{-}\mathsf{Dss}(1^k)$, the optimized $q2$-signature scheme derived by applying Construction 5.1 and the optimization explained above, is existentially unforgeable under adaptive chosen message attacks.*

Proof. Regarding Lemma A.2, note that by removing duplicate information from the signature, we do not affect the ability to extract in any way, and thus the probability of success of the adversary remains exactly the same. Thus Lemma A.2 also holds for $opt\text{-}q2\text{-}\mathsf{Dss}(1^k)$.

For Lemma A.4, the arguments are exactly the same with the exception that the probability of abort of $\mathcal{A}$ may now be different, but nevertheless, still negligible. Indeed, the proof of Lemma A.4 uses the fact that the first signature element σ_1 only takes a given value with negligible probability. This follows from the fact that the commitment scheme has big enough output entropy – and thereby also takes a given value with negligible probability. In the case of $opt\text{-}q2\text{-}\mathsf{Dss}(1^k)$, this statement follows from the same property of the commitment scheme but also from the randomness of the RO that we used to model the hash function $\mathcal{H}$. Hence, the proof of Lemma A.4 also goes through for $opt\text{-}q2\text{-}\mathsf{Dss}(1^k)$.

Now, the rest of the proof proceeds exactly the same as in Theorem 5.2. □

Based on this corollary we can now prove Theorem 10.1.

Proof (of Theorem 10.1). Towards a contradiction, assume there exists an adversary $\mathcal{A}$ that wins the $\mathsf{EU\text{-}CMA}$ game against MQDSS with non-negligible success probability.

We show that this implies the existence of an oracle machine $\mathcal{M}^{\mathcal{A}}$ that solves the $\mathcal{MQ}$ problem, breaks a property of one of the commitment schemes, or distinguishes the outputs of one of the pseudorandom generators from random. We first define a series of games and argue that the difference in success probability of $\mathcal{A}$ between these games is negligible. We assume that $\mathcal{M}$ runs $\mathcal{A}$ in these games.

Game 0: Is the $\mathsf{EU\text{-}CMA}$ game for MQDSS.
Game 1: Is Game 0 with the difference that $\mathcal{M}$ replaces the outputs of $\mathrm{PRG}_{\mathbf{rte}}$ by random bit strings.
Game 2: Is Game 1 with the difference that $\mathcal{M}$ replaces the outputs of $\mathrm{PRG}_{\mathsf{sk}}$ by random bit strings.
Game 3: Is Game 2 with the difference that $\mathcal{M}$ replaces the outputs of $\mathrm{PRG}_{\mathbf{s}}$ by random bit strings.
Game 4: Is Game 3 with the difference that $\mathcal{M}$ replaces the outputs of PRG_{ρ} by random bit strings.
Game 5: Is Game 4 with the difference that $\mathcal{M}$ takes as additional input a random equation system $\mathbf{F}$. $\mathcal{M}$ simulates XOF_F towards $\mathcal{A}$, programming XOF_F such that it returns the coefficients representing $\mathbf{F}$ upon input of S_F and uniformly random values on any other input.

Per assumption, $\mathcal{A}$ wins Game 0 with non-negligible success probability. Let's call this ϵ. If the difference in $\mathcal{A}$'s success probability playing Game 0 or Game 1 was non-negligible, we could use $\mathcal{A}$ to distinguish the outputs of $\mathrm{PRG}_{\mathbf{rte}}$ from random. The same argument applies for the difference between Game 1 and Game 2, between Game 2 and Game 3 and between Game 3 and Game 4. Finally, the output distribution of XOF_F in Game 5 is the same as in previous games. Hence, there is no difference for $\mathcal{A}$ between Game 4 and Game 5. Accordingly, $\mathcal{A}$'s success probability in these two games is equal.

Now, Game 5 is exactly the $\mathsf{EU\text{-}CMA}$ game for the optimized *opt*-$q2$ signature scheme that is derived from $\mathcal{MQ}$-IDS, the 5-pass IDS from [43].

Next, recall that under the assumption of intractability of the $\mathcal{MQ}$ problem on average and assuming computationally binding and computationally hiding properties of Com_0 and Com_1, $\mathcal{MQ}$-IDS is KOW (c.f. Theorem 3.1), is computationally HVZK (c.f. Theorem 4) and has a $q2$-extractor (c.f. Theorem 7). We can now apply Corollary A.6 on $\mathcal{MQ}$-IDS, and obtain that the *opt*-$q2$ signature scheme derived from $\mathcal{MQ}$-IDS is EU-CMA secure. This is a contradiction to the assumption that $\mathcal{A}$ wins Game 5 with non-negligible probability. □