CROSS

Codes and Restricted Objects Signature Scheme

Security Details

Version 2.2 - July 31, 2025

- Marco Baldi, Polytechnic University of Marche, Department of Information Engineering
- Alessandro Barenghi, Politecnico di Milano, Department of Electronics, Information and Bioengineering
- Michele Battagliola, Università degli Studi di Trento, Department of Mathematics
- Sebastian Bitzer, Technical University of Munich, Institute for Communications Engineering
- Marco Gianvecchio, Politecnico di Milano, Department of Electronics, Information and Bioengineering
- Patrick Karl, Technical University of Munich, Chair of Security in Information Technology
- Felice Manganiello, Clemson University, School of Mathematical and Statistical Sciences
- Alessio Pavoni, Polytechnic University of Marche, Department of Information Engineering
- Gerardo Pelosi, Politecnico di Milano, Department of Electronics, Information and Bioengineering
- Federico Pintore, Università di Trento, Department of Mathematics
- Paolo Santini, Polytechnic University of Marche, Department of Information Engineering
- Jonas Schupp, Technical University of Munich, Chair of Security in Information Technology
- Edoardo Signorini, Telsy S.p.A.
- Freeman Slaughter, Clemson University, School of Mathematical and Statistical Sciences
- Antonia Wachter-Zeh, Technical University of Munich, Institute for Communications Engineering
- Violetta Weger, Technical University of Munich, Department of Mathematics

Contents

1 Introduction 2

2 Notation 2

3 Hardness Assumption: Restricted Decoding 3

3.1 Solvers for R-SDP . . . 3

3.1.1 Collision Search-based Solver . . . 4

3.1.2 Representation Technique-based Solver . . . 5

3.2 Solvers for R-SDP(G) . . . 9

4 Proof of Security 14

4.1 Basics on ZK protocols . . . 14

4.2 Security of the Protocol . . . 15

4.3 Signature Scheme Security in General . . . 18

4.4 Security of the Signature Scheme . . . 19

4.5 Forgery Attacks . . . 20

4.6 Finite Regime Cost of a Forgery . . . 24

5 Bibliography 25

A Proof of NP-Completeness 26

1 Introduction

This document provides a thorough security analysis of CROSS. It aims to provide a detailed understanding and to stimulate further research. The security of CROSS relies on two assumptions: The hardness of restricted decoding and the security of a ZK protocol. This security guide is structured accordingly.

Hardness of restricted decoding: Section 3 analyzes the computational cost of solving restricted decoding problems. Section 3.1 provides a detailed analysis of state-of-the-art solvers, Section 3.2 is dedicated to the subgroup variant of the problem. Finally, it is shown that the parameters of CROSS attain the corresponding security levels with respect to these attacks.

Security proof and forgeries: Section 4.4 shows that CROSS is EUF-CMA secure and analyses the security loss due to the Fiat-Shamir transform. Section 4.5 describes two forgery attacks against the underlying identification protocol and the signature obtained by applying the Fiat-Shamir transform. The latter is based on a novel attack from [5] and its cost estimate is used for the selection of CROSS parameters.

Artifacts: Scripts for reproducing the attack costs, including all presented figures and tables are available at `https://www.cross-crypto.com/resources`.

2 Notation

The security guide uses the same notation as the main specification. Vectors and matrices are denoted as bold lowercase and bold capital letters, respectively. Furthermore, elements, vectors, and matrices in F_z are overlined. Table 1 provides a quick reference for the remaining notation used.

Table 1: Mathematical notation

Symbol	Meaning
p, z	Prime numbers, $z < p$
$\mathbb{F}_p$	Finite field with p elements
$\mathbb{F}_p^*$	Multiplicative group, $\mathbb{F}_p \setminus \{0\}$
$\mathbb{F}_z$	Finite field with z elements
$\mathbb{E}$	Cyclic subgroup of $(\mathbb{F}_p^*, \cdot)$, with generator g of order z
$\mathbb{E}_0$	$\mathbb{E} \cup \{0\}$
$\star$	Component-wise multiplication
G	Subgroup of $(\mathbb{E}^n, \star)$ of size z^m
Id_ℓ	Identity matrix of size $\ell \times \ell$
n	Code length and length of restricted vectors
m	Size of the subgroup G is z^m, $m < n$
k	Code dimension, with $k < n$
λ	Security parameter
t	Number of rounds
w	Weight of the second challenge
$\mathcal{B}_{(t,w)}$	Hamming sphere of vectors in $\mathbb{F}_2^t$ with radius w
$\mathbb{P}(A)$	Probability of the event A
$\mathtt{poly}(\lvert x\rvert)$	Polynomial-time complexity $2^{O(\log(\lvert x\rvert))}$

3 Hardness Assumption: Restricted Decoding

CROSS utilizes restricted decoding problems of the following form.

Problem 1. ***Restricted Syndrome Decoding Problem (R-SDP)***
Let $g \in \mathbb{F}_p^*$ be of order z, $\mathbf{H} \in \mathbb{F}_p^{(n-k)\times n}$, $\mathbf{s} \in \mathbb{F}_p^{n-k}$, and $\mathbb{E} = \{g^i \mid i \in \{1, \dots, z\}\} \subset \mathbb{F}_p^*$.
Does there exist a vector $\mathbf{e} \in \mathbb{E}^n$ such that $\mathbf{eH}^\top = \mathbf{s}$?Po

It can be shown that R-SDP is NP-complete.

Theorem 2. The R-SDP (Problem 1) is NP-complete.

For the proof, we refer the interested reader to Appendix A. In fact, this result is not surprising since R-SDP is tightly connected to other well-known decoding problems. In particular, for $z = p - 1$, we recover syndrome decoding with full weight. For $z = 2$, R-SDP is related to the subset sum problem over finite fields. CROSS uses R-SDP with $p = 127$ and $z = 7$.

R-SDP can be generalized by considering a subgroup $(G, \star) \leq (\mathbb{E}^n, \star)$, where $\star$ denotes component-wise multiplication. Let $\mathbf{a}_1, \dots, \mathbf{a}_m \in \mathbb{E}^n$. For some $m \leq n$, we define the subgroup as

$$G = \langle \mathbf{a}_1, \dots, \mathbf{a}_m \rangle = \left\{ \star_{i=1}^m \mathbf{a}_i^{\overline{u}_i} \mid \overline{u}_i \in \mathbb{F}_z \right\}.$$

A variant of CROSS uses this generalization, to which we refer as R-SDP(G).

Problem 3. ***Restricted Syndrome Decoding Problem with Subgroup (R-SDP(G))***
Let $G = \langle \mathbf{a}_1, \dots, \mathbf{a}_m \rangle$, for $\mathbf{a}_i \in \mathbb{E}^n$, $\mathbf{H} \in \mathbb{F}_p^{(n-k)\times n}$, and $\mathbf{s} \in \mathbb{F}_p^{n-k}$.
Does there exist a vector $\mathbf{e} \in G$ with $\mathbf{eH}^\top = \mathbf{s}$?

For $m = n$, R-SDP (Problem 1) is recovered. CROSS uses R-SDP(G) with $m < n$, $p = 509$ and $z = 127$. Technically, CROSS relies on the hardness of the following computational version with a planted solution. Note that $m = n$ recovers R-SDP.

Problem 4. ***Computational R-SDP and R-SDP(G)***
Let $G = \langle \mathbf{a}_1, \dots, \mathbf{a}_m \rangle$ with $\mathbf{a}_i \in \mathbb{E}^n$. Further, let $\mathbf{H} \in \mathbb{F}_p^{(n-k)\times n}$, and $\mathbf{s} = \mathbf{e}^*\mathbf{H}^\top \in \mathbb{F}_p^{n-k}$ with $\mathbf{e}^* \in G$.
Find a vector $\mathbf{e}$ such that $\mathbf{e} \in G$ and $\mathbf{eH}^\top = \mathbf{s}$.

Expected number of solutions. The number of valid solutions is crucial for the computational hardness of solving R-SDP and R-SDP(G). We quantify the expected number of solutions in the following theorem, for which the case of R-SDP can be recovered by setting $G = \mathbb{E}^n$, i.e., $m = n$.

Theorem 5 ([22])**.** Let an R-SDP instance be given with $G = \langle \mathbf{a}_1, \dots \mathbf{a}_m \rangle$, $\mathbf{H} \in \mathbb{F}_p^{(n-k)\times n}$, and $\mathbf{s} = \mathbf{e}^*\mathbf{H}^\top$, where $\mathbf{e}^* \in G$. Then, the number of solutions of this instance is a random variable over the choice of $\mathbf{H}$ with expected value

$$\frac{z^m - 1}{p^{n-k}} + 1.$$

Proof. We have at least one solution, $\mathbf{e}^*$. Thus, the probability of $\mathbf{e} = \mathbf{e}^*$ to be a solution is one, i.e., $\mathbb{P}(\mathbf{eH}^\top = \mathbf{s}) = 1$. For any other restricted vector $\mathbf{e} \in G \setminus \{\mathbf{e}^*\}$, this probability is $\mathbb{P}(\mathbf{eH}^\top = \mathbf{s}) = p^{k-n}$, assuming that $\mathbf{H}$ is chosen uniform at random. Due to the linearity of the expectation, the expected number of solutions is given by

$$\sum_{\mathbf{e} \in G} \mathbb{P}(\mathbf{eH}^\top = \mathbf{s}) = \frac{z^m - 1}{p^{n-k}} + 1.$$

□

3.1 Solvers for R-SDP

This section discusses state-of-the-art solvers for R-SDP and derives estimates for their computational complexity. The presented solvers lie at the intersection of Information Set Decoding (ISD) [7, 8, 11, 26] and the best-known solvers for the subset sum problem [6, 16]. These solvers have been adapted to

R-SDP in [10] for the particular case of $z \in \{2,4,6\}$ and in [3] for arbitrary values of z. Here, we focus on the parameter choice of CROSS, which uses

$$p = 127 \text{ and } z = 7, \text{ that is } \mathbb{E} = \{1,2,4,8,16,32,64\}.$$

This choice avoids weaker instances, such as $\mathbb{E}$ being a subfield; see [10, 21].

Combinatorial attacks: This section focuses on *combinatorial* solvers since those yield the best performances for R-SDP. A detailed study of *algebraic* solvers can be found in [9], which comes to the conclusion that Gröbner bases, even using hybrid techniques, are not competitive.

Quasi-systematic form: All modern ISD algorithms use a parity-check matrix in quasi-systematic form [13]. To bring an arbitrary parity-check matrix $\mathbf{H}$ into this form, one applies a permutation to its columns by computing $\mathbf{HP}$ with $\mathbf{P}$ a $n \times n$ permutation matrix. Using partial Gaussian elimination, an invertible $\mathbf{U} \in \mathbb{F}_p^{(n-k)\times(n-k)}$ is determined that transforms $\mathbf{HP}$ into

$$\mathbf{UHP} = \mathbf{H}' = \begin{pmatrix} \mathrm{Id}_{n-k-\ell} & \mathbf{H}_1 \\ \mathbf{0} & \mathbf{H}_2 \end{pmatrix},$$

where $\mathbf{H}_1 \in \mathbb{F}_p^{(n-k-\ell)\times(k+\ell)}$ and $\mathbf{H}_2 \in \mathbb{F}_p^{\ell\times(k+\ell)}$. The same transformation is applied to the syndrome, i.e., one computes

$$\mathbf{Us} = \mathbf{s}' = \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix},$$

where $\mathbf{s}_1 \in \mathbb{F}_p^{n-k-\ell}$ and $\mathbf{s}_2 \in \mathbb{F}_p^{\ell}$. This inherently splits the unknown error vector $\mathbf{e}$ into $\mathbf{e}_1 \in \mathbb{E}^{n-k-\ell}$ and $\mathbf{e}_2 \in \mathbb{E}^{k+\ell}$ with $\mathbf{eP}^\top = (\mathbf{e}_1, \mathbf{e}_2)$. That is, we get the system of two equations

$$\begin{aligned} \mathbf{e}_1 + \mathbf{e}_2\mathbf{H}_1^\top &= \mathbf{s}_1 \text{ and} \\ \mathbf{e}_2\mathbf{H}_2^\top &= \mathbf{s}_2. \end{aligned}$$

To solve this system, one enumerates solutions $\mathbf{e}_2$ of the second equation $\mathbf{e}_2\mathbf{H}_2^\top = \mathbf{s}_2$ and checks for each one if the remaining $\mathbf{e}_1 = \mathbf{s}_1 - \mathbf{e}_2\mathbf{H}_1^\top$ completes it to a valid, i.e., restricted, solution. In the following, we discuss two methods for enumerating candidates $\mathbf{e}_2$.

3.1.1 Collision Search-based Solver

A meet-in-the-middle strategy can be used to enumerate the solutions. This approach was applied to hard knapsacks by Horowitz and Sahni [15] and adopted for the syndrome decoding problem by Stern [26] and Dumer [11]. For the adaption to R-SDP, we define the lists

$$\begin{aligned} \mathcal{L}_a &:= \left\{ (\mathbf{e}_a, \quad (\mathbf{e}_a, \mathbf{0})\mathbf{H}_2^\top) \mid \mathbf{e}_a \in \mathbb{E}^{\lfloor \frac{k+\ell}{2} \rfloor} \right\} \text{ and} \\ \mathcal{L}_b &:= \left\{ (\mathbf{e}_b, \mathbf{s}_2 - (\mathbf{0}, \mathbf{e}_b)\mathbf{H}_2^\top) \mid \mathbf{e}_b \in \mathbb{E}^{\lceil \frac{k+\ell}{2} \rceil} \right\}, \end{aligned}$$

which contain $L_a = |\mathcal{L}_a| = z^{\lfloor \frac{k+\ell}{2} \rfloor}$ and $L_b = |\mathcal{L}_b| = z^{\lceil \frac{k+\ell}{2} \rceil}$ elements, respectively. We refer to the second entry of each tuple as the *label* of the list element. Using a hashmap or sorting, one finds collisions between the labels of $\mathcal{L}_a$ and $\mathcal{L}_b$, each of which yields a $\mathbf{e}_2 = (\mathbf{e}_a, \mathbf{e}_b)$ that satisfy $\mathbf{e}_2\mathbf{H}_2^\top = \mathbf{s}_2$.

As described previously, each such partial solution is then checked if it extends to a solution of the original problem.

Theorem 6. The discussed collision-based solver COLL requires at least

$$M_{\mathsf{COLL}}(p,n,k,z) = L_a \cdot \left\lfloor \frac{k+\ell}{2} \right\rfloor \cdot \log_2(z).$$

bits of memory. The required number of binary operations is bounded from below as

$$C_{\mathsf{COLL}} \geq \min_{0\leq\ell\leq n-k} \frac{C_a + C_b + C_c}{1+(z^n-1)p^{k-n}} \log_2(M_{\mathsf{COLL}}(p,n,k,z)),$$

where C_a, C_b and C_c are bounded as

$$\begin{aligned} C_a &\geq L_a \cdot \left(\left\lfloor \frac{k+\ell}{2} \right\rfloor \cdot \log_2(z) + \ell \cdot \log_2(p)\right), \\ C_b &\geq L_b \cdot \left(\left\lceil \frac{k+\ell}{2} \right\rceil \cdot \log_2(z) + \ell \cdot \log_2(p)\right), \\ C_c &\geq L_a \cdot L_b \cdot p^{-\ell} \cdot (k+\ell)\log_2(p). \end{aligned}$$

Proof. To perform the collision search, the algorithm has to store the smaller list among $\mathcal{L}_a$ and $\mathcal{L}_b$. Since this list contains dense vectors of length $\left\lfloor \frac{k+\ell}{2} \right\rfloor$ with entries in $\mathbb{E}$, at least $\left\lfloor \frac{k+\ell}{2} \right\rfloor \log_2(z)$ bits are required per list element. This gives the bound on the memory cost.

Let us consider the solver's time complexity, estimated as the cost of finding a particular solution divided by the number of solutions. According to Theorem 5, the average number of solutions is given by $1 + (z^n - 1)p^{k-n}$.

First, one computes for each error vector $\mathbf{e}_a$ associated with list $\mathcal{L}_a$ the corresponding syndrome $(\mathbf{e}_a, \mathbf{0})\mathbf{H}_2^\top$. The error vectors have a size of $\left\lfloor \frac{k+\ell}{2} \right\rfloor \cdot \log_2(z)$ bit and the syndromes a size of $\ell \cdot \log_2(p)$ bit. Hence, we estimate the required number of binary operations as $L_a \cdot \left(\left\lfloor \frac{k+\ell}{2} \right\rfloor \cdot \log_2(z) + \ell \cdot \log_2(p)\right)$.

Next, the syndromes $\mathbf{s}_2 - (\mathbf{0}, \mathbf{e}_b)\mathbf{H}_2^\top$ of the error vectors $\mathbf{e}_b$ associated with list $\mathcal{L}_b$ are calculated. Again, due to the size of the objects, this requires at least $L_b \left(\left\lceil \frac{k+\ell}{2} \right\rceil \cdot \log_2(z) + \ell \cdot \log_2(p)\right)$ binary operations.

Solutions $\mathbf{e}_2$ of the small instance are obtained by performing a collision search, i.e., $(\mathbf{e}_a, \mathbf{0})\mathbf{H}_2^\top = \mathbf{s}_2 - (\mathbf{e}_b, \mathbf{0})\mathbf{H}_2^\top$. On average, $L_a \cdot L_b \cdot p^{-\ell}$ collisions are found. For each collision, one checks whether $\mathbf{e}_2 = (\mathbf{e}_a, \mathbf{e}_b)$ extends to a solution of the original problem. Even for false positives, one has to calculate at least one syndrome symbol of the complete instance, which is the sum of $k + \ell$ elements of $\mathbb{F}_p$. Hence, this step requires at least $L_a \cdot L_b \cdot p^{-\ell} \cdot (k+\ell)\log_2(p)$ binary operations.

Finally, the memory access cost is modeled with the conservative logarithmic cost model [2, 12]. That is, the cost per iteration is increased by a factor $\log_2(M_{\mathsf{COLL}}(p, n, k, z))$. □

3.1.2 Representation Technique-based Solver

Overview: We now analyze a more elaborate multi-level algorithm inspired by [6, 7, 14, 16]. This algorithm uses representations from a sum partition instead of the above set partition. That is, one writes the solution to the smaller instance as $\mathbf{e}_2 = \mathbf{e}_a + \mathbf{e}_b$, where $\mathbf{e}_a$ and $\mathbf{e}_b$ are suitably chosen vectors of length $k + \ell$, the supports of which may overlap. Then, there are multiple pairs $(\mathbf{e}_a, \mathbf{e}_b)$, which follow a chosen distribution and sum to $\mathbf{e}_2$. Since it is sufficient to obtain a single copy of $\mathbf{e}_2$ to solve the problem, it is sufficient to enumerate only a fraction of all possible pairs $(\mathbf{e}_a, \mathbf{e}_b)$. For a detailed introduction to the representation technique, the reader is referred to [22].

Picking the ambient space: To minimize the number of vectors that must be enumerated and, hence, the computational complexity, we tailor the representation technique to the restricted case. Similar to [6], it can be beneficial to construct lists of vectors with entries that are not in $\mathbb{E}_0$ but in $\mathbb{D}$, where $\mathbb{D} \subseteq \{a - b \mid a, b \in \mathbb{E}\} \setminus \mathbb{E}_0$ is a carefully chosen set, which allows for an increased number of representations. We denote by $z_\mathbb{D}$ the size of the chosen $\mathbb{D}$.

Example 7. CROSS utilizes $p = 127$ and $\mathbb{E} = \{1, 2, 4, 8, 16, 32, 64\}$. We pick

$$\mathbb{D} = \{a - b \mid a, b \in \mathbb{E}\} \setminus \mathbb{E}_0 = \{2^{i_1} - 2^{i_1 + i_2} \mid i_1 \in \{0, \ldots, 6\}, i_2 \in \{1, \ldots, 5\}\},$$

which contains $z_\mathbb{D} = 35$ elements.

Counting representations: To determine the number of representations of an error vector as a sum of vectors in $(\mathbb{E}_0 \cup \mathbb{D})^{k+\ell}$, we quantify the additive structure of $\mathbb{E}$ and $\mathbb{D}$ in the following. For this, we determine the number of possibilities to write an element $a \in \mathbb{E}$ as $b + c$ with $b, c \in \mathbb{E}$ and the number of possibilities to write it as $b + \tilde{c}$ with $b \in \mathbb{E}, \tilde{c} \in \mathbb{D}$. These quantities are denoted by

$$\begin{aligned} \alpha_\mathbb{E}(a) &:= |\{b \in \mathbb{E} \mid \exists c \in \mathbb{E} : b + c = a\}|, \\ \alpha_\mathbb{D}(a) &:= |\{b \in \mathbb{E} \mid \exists \tilde{c} \in \mathbb{D} : b + \tilde{c} = a\}|. \end{aligned}$$

For the chosen $\mathbb{E}$ in CROSS, we note that these quantities do not depend on the choice of $a \in \mathbb{E}$. Thus, we simply write $\alpha_\mathbb{E}$ and $\alpha_\mathbb{D}$.

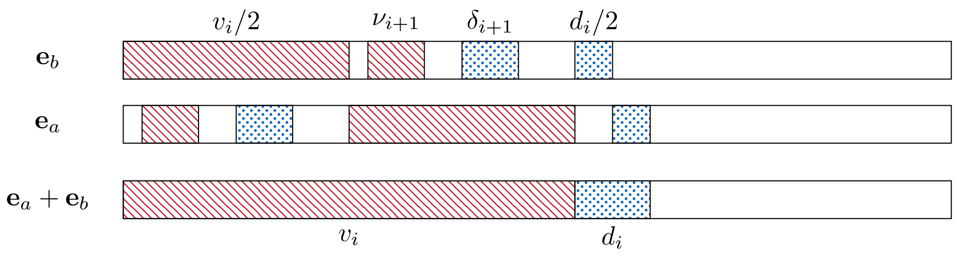

Figure 1: Illustration for counting the number of representations on level i.
Entries in $\mathbb{E}$ are drawn in red, entries in $\mathbb{D}$ are drawn in blue.

Example 8. Let $\mathbb{E} = \{1, 2, 4, 8, 16, 32, 64\}$ and $\mathbb{D}$ be as chosen in Example 7. Then, $\alpha_\mathbb{E} = 1$ since $2a \in \mathbb{E}$ for all $e \in \mathbb{E}$. For any element $a \in \mathbb{E}$ there exists five elements $\tilde{c} \in \mathbb{D}$ such that $\tilde{c} + a \in \mathbb{E}$; hence, $\alpha_\mathbb{D} = 5$. More generally, there is a $\widetilde{\mathbb{D}}$ of size $z_{\widetilde{\mathbb{D}}} = z \cdot s$ with $s \in \{1, \ldots, 5\}$ such that $\alpha_{\widetilde{\mathbb{D}}} = s$.

We now have all the necessary preliminaries to count the number of representations, as stated in the following lemma.

Lemma 9. Let $\mathbf{e} \in (\mathbb{E}_0 \cup \mathbb{D})^{k+\ell}$ have v_i entries from $\mathbb{E}$ and d_i entries from $\mathbb{D}$. Further, we let $\nu_{i+1} = v_{i+1} - \frac{v_i}{2}$ and $\delta_{i+1} = d_{i+1} - \frac{d_i}{2}$. Then, there are

$$r = \binom{v_i}{v_{i+1}}\binom{v_{i+1}}{2\nu_{i+1}}\alpha_\mathbb{E}^{2\nu_{i+1}} \cdot \binom{v_i/2 - \nu_{i+1}}{\delta_{i+1}}^2 \cdot \alpha_\mathbb{D}^{2\delta_{i+1}}\binom{d_i}{d_i/2}$$

possibilities for picking $\mathbf{e}_a, \mathbf{e}_b \in (\mathbb{E}_0 \cup \mathbb{D})^{k+\ell}$ such that $\mathbf{e}_a, \mathbf{e}_b$ each have v_{i+1} entries in $\mathbb{E}$, d_{i+1} entries in $\mathbb{D}$, and $\mathbf{e}_a + \mathbf{e}_b = \mathbf{e}$.

Proof. The counting of the number of representations is depicted in Figure 1. For the v_i entries of $\mathbf{e}$ living in $\mathbb{E}$, we choose $v_{i+1} = v_i/2 + \nu_{i+1}$ entries in $\mathbf{e}_a$ and distribute inside these v_{i+1} entries the $2\nu_{i+1}$ overlaps with entries of $\mathbf{e}_b$ in $\mathbb{E}$. By definition, there are $\alpha_\mathbb{E}^{2\nu_{i+1}}$ ways of choosing the $2\nu_{i+1}$ entries.

Then, out of the non-selected $v_i - v_{i+1}$ entries of $\mathbf{e}$ in $\mathbb{E}$, we choose δ_{i+1} many entries of $\mathbf{e}_a$ for overlaps with entries in $\mathbf{e}^{(2)}$. This step is repeated for $\mathbf{e}_b$, for which we pick δ_{i+1} entries of $\mathbf{e}_a$ in $\mathbb{E}$. By definition, there are again $\alpha_\mathbb{D}^{2\delta_{i+1}}$ choices for these entries.

Finally, we split the d_i entries of $\mathbf{e}$ living in $\mathbb{D}$ into $d_i/2$ entries of $\mathbf{e}_a$ living in $\mathbb{D}$, which then also fixes the remaining $d_i/2$ entries of $\mathbf{e}_b$ in $\mathbb{D}$. □

Multi-level solver: We now describe how the multi-level algorithm proceeds in the case of four levels. We also tried more levels; however, increasing the number of levels further did not yield an improved finite regime performance. The compositions of the levels are connected via

$$\begin{aligned} &v_0 = k + \ell, \quad &&v_1 = v_0/2 + \nu_1, \quad &&v_2 = v_1/2 + \nu_2, \quad &&v_3 = v_2/2, \\ &d_0 = 0, &&d_1 = d_0/2 + \delta_1, &&d_2 = d_1/2 + \delta_2, &&d_3 = d_2/2, \end{aligned}$$

where $\ell, \nu_1, \nu_2, \delta_1$ and δ_2 are internal parameters which can be optimized. The parameter ℓ denotes the redundancy of the small instance due to the partial Gaussian elimination, and ν_i and δ_i correspond to the "overlapping" number of entries in $\mathbb{E}$, respectively in $\mathbb{D}$ on level i.

Then, according to Lemma 9, the number of representations for level 1, i.e., r_1, and the number of representation for level 0, i.e., r_0, are given by

$$\begin{aligned} r_1 &= \binom{v_1}{v_2}\binom{v_2}{2\nu_2}\alpha_\mathbb{E}^{2\nu_2}\binom{v_1 - v_2}{\delta_2}^2 \alpha_\mathbb{D}^{2\delta_2}\binom{d_1}{d_1/2}, \\ r_0 &= \binom{v_0}{v_1}\binom{v_1}{2\nu_1}\alpha_\mathbb{E}^{2\nu_1}\binom{v_0 - v_1}{\delta_1}^2 \alpha_\mathbb{D}^{2\delta_1}. \end{aligned}$$

The algorithm operates as follows:

Level 3: The algorithm prepares the base lists $\mathcal{L}_3$. The elements of the base lists are vectors of length $\frac{k+\ell}{2}$ which contain v_3 entries of $\mathbb{E}$ and d_3 entries of $\mathbb{D}$. Each base list has the same size, being

$$L_3 = \binom{(k+\ell)/2}{v_3, d_3} z^{v_3} z_{\mathbb{D}}^{d_3},$$

where $\binom{(k+\ell)/2}{v_3,d_3} = \binom{(k+\ell)/2}{v_3+d_3} \cdot \binom{v_3+d_3}{v_3}$ denotes the trinomial coefficient.

Level 2: Two base lists are merged into a list by performing a concatenation merge on ℓ_1 symbols. We refer to the resulting list as $\mathcal{L}_2$, which contains vectors of length $k+\ell$ with v_2 entries of $\mathbb{E}$ and d_2 entries of $\mathbb{D}$. The lists $\mathcal{L}_2$ have sizes

$$L_2 = \binom{k+\ell}{v_2, d_2} z^{v_2} z_{\mathbb{D}}^{d_2} p^{-\ell_1},$$

where $\ell_1 = \log_p(r_1)$ guarantees that one representation of the final solution in $\mathcal{L}_2$ survives the merge on average.

Level 1: The algorithm creates lists by performing a representation merge of two level-2 lists on ℓ_0 syndrome symbols. We refer to the resulting list as $\mathcal{L}_1$, which contains vectors of length $k+\ell$ with v_1 entries of $\mathbb{E}$ and d_1 entries of $\mathbb{D}$. The lists $\mathcal{L}_1$ have size

$$L_1 = \binom{k+\ell}{v_1, d_1} z^{v_1} z_{\mathbb{D}}^{d_1} p^{-\ell_0-\ell_1},$$

where $\ell_0 = \log_p(r_0) - \ell_1$ guarantees that one representation of the final solution in $\mathcal{L}_1$ survives the merge on average.

Level 0: A final representation merge on the remaining $\ell - \ell_1 - \ell_0$ syndrome symbols gives a solution of the small instance, i.e., vectors $\mathbf{e}_2$ of length $k+\ell$ with entries solely from $\mathbb{E}$ that satisfy $\mathbf{e}_2\mathbf{H}_2^\top = \mathbf{s}_2$.

The following theorem summarizes the computational cost of the presented solver.

Theorem 10. The presented representation-based solver REPR uses at least

$$M_{\mathsf{REPR}}(p,n,k,z) = \max_{i\in\{3,2,1\}} \{L_i(v_i \log_2(z) + d_i \log_2(z_{\mathbb{D}}))\}.$$

bits of memory. The computational complexity of the algorithm can be bounded from below as

$$C_{\mathsf{REPR}}(p,n,k,z) = \min_{\ell,\nu_1,\nu_2,\delta_1,\delta_2} \left\{ \frac{C_3+C_2+C_1+C_0}{1+(z^n-1)p^{k-n}} \log_2(M_{\mathsf{REPR}}(p,n,k,z)) \right\},$$

where C_i denotes the cost associated with level i, which are given as

$$\begin{aligned}
C_3 &\geq 2 \cdot L_3(\ell_1 \log_2(p) + v_3 \log_2(z) + d_3 \log_2(z_{\mathbb{D}})),\\
C_2 &\geq 2 \cdot L_2(\ell_0 \log_2(p) + v_2 \log_2(z) + d_2 \log_2(z_{\mathbb{D}})),\\
C_1 &\geq 2 \cdot (L_2)^2 \cdot p^{-\ell_0} \log_2(p),\\
C_0 &\geq (L_1)^2 \cdot p^{-(\ell-\ell_0-\ell_1)} \log_2(p).
\end{aligned}$$

Proof. To perform the collision search, the solver stores at least one of the lists on levels 1, 2, and 3. The final list does not need to be stored, as it can be checked on the fly. On level 3, elements of the base lists $\mathcal{L}_3$ require at least $(v_3 \log_2(z) + d_3 \log_2(z_{\mathbb{D}}))$ bits of memory. Similarly, each element of $\mathcal{L}_2$ requires at least $(v_2 \log_2(z) + d_2 \log_2(z_{\mathbb{D}}))$ bits, and each element of $\mathcal{L}_1$ requires at least $(v_1 \log_2(z) + d_1 \log_2(z_{\mathbb{D}}))$ bits. This gives the bound on the memory cost $M_{\mathsf{REPR}}(p,n,k,z)$.

Let us now consider the time complexity of the algorithm:

Level 3: The base lists $\mathcal{L}_3$ are constructed. Similar to the collision-based solver, one constructs at least two such lists to perform the first concatenation merge. For each element, which has size of at least $(v_3 \log_2(z) + d_3 \log_2(z_{\mathbb{D}}))$ bits, one calculates a partial syndrome in $\mathbb{F}_p^{\ell_1}$. This gives the lower bound on the cost C_3.

Level 2: A concatenation merge is performed on the base lists, resulting in $L_2 = (L_3)^2 \cdot p^{-\ell_1}$ collisions. For each collision, one obtains an error vector, which has a size of at least $(v_2 \log_2(z) + d_2 \log_2(z_{\mathbb{D}}))$ bits, and calculates a partial syndrome in $\mathbb{F}_p^{\ell_0}$. Again, this step has to be performed at least twice to continue to lower levels. Hence, we obtain the bound on C_2.

Level 1: List $\mathcal{L}_2$ are merged on ℓ_0 syndrome symbols. This representation merge yields on average $(L_2)^2 p^{\ell_0}$ collisions. Taking into account early abort techniques [8], we conservatively estimate the cost per collision as a single field addition: at least one entry of the error vector has to be added to determine whether the sum of the vectors is well-formed. Considering that this step needs to be performed twice, we obtain the lower bound on C_1.

Level 0: One performs a final representation merge between two lists $\mathcal{L}_1$ on the remaining $\ell - \ell_0 - \ell_1$ syndrome symbols of the small instance. This representation merge yields on average $(L_1)^2 \cdot p^{-(\ell-\ell_0-\ell_1)}$ collisions. Again, we conservatively estimate the cost per collision as a single field addition.

Finally, the memory access cost is modeled with the conservative logarithmic cost model [2, 12]. That is, the cost per iteration is increased by a factor $\log_2(M_{\mathsf{REPR}}(p, n, k, z))$. □

A comment on success probability: The parameters ℓ_0 and ℓ_1 are selected such that the average number of surviving representations is one. Nevertheless, there is a non-zero probability that no representation survives. These probabilities can be compensated by performing a (small) number of iterations, the cost of which is disregarded in the presented analysis.

Shifting $\mathbb{E}$: An R-SDP instance can be transformed into an instance with a modified restriction. Denote the columns of $\mathbf{H}$ as $\mathbf{h}_0, \dots, \mathbf{h}_{n-1}$, set $\mathbf{x} = (x, \dots, x) \in \mathbb{F}_p^n$, and define

$$\widetilde{\mathbf{H}} = \left(\mathbf{h}_0 \cdot g^{i_0}, \quad \dots, \quad \mathbf{h}_{n-1} \cdot g^{i_{n-1}}\right) \quad \text{and} \quad \tilde{\mathbf{s}} = \mathbf{s} + \mathbf{x}\mathbf{H}^\top.$$

Then, $\tilde{\mathbf{e}} = \mathbf{e} \star (g^{i_0}, \dots, g^{i_{n-1}}) - \mathbf{x}$ is a solution to the modified R-SDP instance with parity-check matrix $\widetilde{\mathbf{H}}$, syndrome $\tilde{\mathbf{s}}$ and restriction $\widetilde{\mathbb{E}} = \{e - x \mid e \in \mathbb{E}\}$. That is, the restricted error vector $\mathbf{e} \in \mathbb{E}^n$ is shifted by x. Therefore, by selecting $x \in \mathbb{E}$, one can construct a modified instance with a solution that contains zeros. We denote set the non-zero entries of the shifted errors as

$$\mathbb{E}_x := \{a - x \mid a \in \mathbb{E}\} \setminus \{0\}.$$

Solving shifted instances: In the following, we elaborate on how shifted instances can be solved using modified variants of the presented solvers. The Hamming weight of the modified instance follows a binomial distribution, i.e., we have

$$\mathbb{P}(\mathrm{wt_H}(\tilde{\mathbf{e}}) = w) = \frac{\binom{n}{w}(z-1)^w}{z^n} \quad \forall w \in \{0, \dots, n\}.$$

In particular, the weight of $\tilde{\mathbf{e}}_2$, i.e., the shifted error restricted to the small instance, is also binomially distributed. Therefore, it is sufficient to enumerate solutions of the small instance with weight v_0, where $0 \le v_0 \le k + \ell$, in order to succeed with probability

$$\mathbb{P}(\mathrm{wt_H}(\tilde{\mathbf{e}}_2) = v_0) = \binom{k+\ell}{v_0}(z-1)^{v_0} z^{-k-\ell}.$$

The total cost of the solver is then the cost of a single iteration divided by the success probability. For the collision-based solver, the required number of iterations compensates the decreased cost per iteration. Therefore, shifting does not provide a speed-up.

The representation-based solver, however, benefits from the zeros since intermediate lists inherently use error vectors that are not of full weight. The cost per iteration can be computed as given in Theorem 10. It remains to analyze the structure of the shifted errors in $\mathbb{E}_x$ and the supplementary elements in $\mathbb{D}_x \subseteq \{b - a \mid a, b \in \mathbb{E}_x\} \setminus (\mathbb{E}_x \cup \{0\})$.

Table 2: Bit-complexity estimates for solvers of R-SDP with parameters as used by CROSS. The restriction is given by $\mathbb{E} = \{1, 2, 4, 8, 16, 32, 64\} \subset \mathbb{F}_{127}$.

Category	Parameters		Solver	Solver parameters								Cost [bit]	
	n	k		ℓ	v_0	v_1	v_2	v_3	d_1	d_2	d_3	Time	Mem.
NIST 1	127	76	COLL	20	96	48						149	141
			REPR	48	128	70	36	18	4	2	1	162	153
			shift REPR	33	72	40	20	10				143	117
NIST 3	187	111	COLL	28	139	69						213	201
			REPR	69	180	104	54	27	4	2	1	229	214
			shift REPR	45	104	56	28	14				207	169
NIST 5	251	150	COLL	38	188	94						281	271
			REPR	90	240	136	70	35	4	2	1	301	275
			shift REPR	68	152	84	42	21				274	234

Example 11. CROSS utilizes the error set $\mathbb{E} = \{1, 2, 4, 8, 16, 32, 64\} \subset \mathbb{F}_{127}$ with size $z = 7$ and additivity $\alpha_{\mathbb{E}} = 1$. Shifting by $x = 1$, the error entries are either zero or lie in the modified error set

$$\mathbb{E}_1 = \{1, 3, 7, 15, 31, 63\}.$$

Unlike its parent $\mathbb{E}$, the modified error set does not possess an additive structure. This holds for shifting by $x = 1$ as well as any other $x \in \mathbb{E}$.

Previous to shifting, $\mathbb{E}$ had a difference set $\mathbb{D}$ of size $z_{\mathbb{D}} = 35$ and additivity $\alpha_{\mathbb{D}} = 5$. After shifting, we obtain $\mathbb{D}_x$ of size $z_{\mathbb{D}_x} = 30$ with $\alpha_{\mathbb{D}_x} = 5$.

Expected security strength: Table 2 presents the computational cost of the presented solvers. The optimized solver parameters are given and can be reproduced using the code available at https://www.cross-crypto.com/resources. The combination of representation technique and shifting yields the best performance for each of the parameter sets, which achieve NIST security categories 1, 3, and 5.

3.2 Solvers for R-SDP(G)

In this section, we extend the discussion of the computational hardness of R-SDP to R-SDP(G). The focus lies on R-SDP(G) as used by CROSS which utilizes

$$p = 509 \text{ and } z = 127, \text{ that is } \mathbb{E} = \{16^i \mid i \in \{1, \dots, 127\}\}.$$

Not considering G: A first naive approach to solving R-SDP(G) would be to enumerate the solutions of the corresponding R-SDP instance, completely dismissing G. Such solutions can be generated, e.g., using the solvers presented in Section 3.1 or an adaption of Wagner's algorithm [27]. Then, for each solution in $\mathbb{E}^n$, it is checked whether $\mathbf{e} \in G$. Note, however, that dismissing G implies that an instance with $1 + (z^n - 1)p^{k-n}$ solutions is solved, out of which only $1 + (z^m - 1)p^{k-n}$ solve the original R-SDP(G) instance (see Theorem 5).

Example 12. For the parameters $p = 509$, $z = 127$, $m = 25$ and $n = 55$, $k = 36$, there are 2^{213} solutions in $\mathbb{E}^n$, while we expect 15.7 solutions in G on average. Hence, on average, 2^{209} solutions in $\mathbb{E}^n$ must be enumerated to find a valid solution for the R-SDP(G) instance.

Collision search with subgroup G: As discussed above, suitably chosen parameters ensure that disregarding G results in costs exceeding the required security levels. In the following, we present a method for incorporating G into a collision-based solver. To this end, let $\overline{\mathbf{H}} \in \mathbb{F}_z^{(n-m)\times n}$ denote a full-rank matrix such that

$$\overline{\mathbf{v}\mathbf{H}}^\top = 0 \iff g^{\overline{\mathbf{v}}} \in G.$$

Thus, $\overline{\mathbf{H}}$ acts like a parity-check matrix for G and, as such, can be used in a collision-based solver.

Both $\mathbf{H}$ and $\overline{\mathbf{H}}$ are transformed into quasi-systematic form. To achieve this, the same column permutation is applied using $\mathbf{P} \in \mathbb{F}_p^{n\times n}$ and $\overline{\mathbf{P}} \in \mathbb{F}_z^{n\times n}$. Through partial Gaussian elimination, invertible matrices $\mathbf{U} \in \mathbb{F}_p^{(n-k)\times(n-k)}$ and $\overline{\mathbf{U}} \in \mathbb{F}_z^{(n-m)\times(n-m)}$ are determined such that

$$\begin{aligned}
&\mathbf{U}\ \mathbf{H}\ \mathbf{P} = \mathbf{H}' = \begin{pmatrix} \mathrm{Id}_{n-k-\ell} & \mathbf{H}_1 \\ 0 & \mathbf{H}_2 \end{pmatrix}, \quad \mathbf{U}\mathbf{s} = \mathbf{s}' = \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix}, \text{ and} \\
&\overline{\mathbf{U}}\ \overline{\mathbf{H}}\ \overline{\mathbf{P}} = \overline{\mathbf{H}}' = \begin{pmatrix} \mathrm{Id}_{n-m-\bar{\ell}} & \overline{\mathbf{H}}_1 \\ 0 & \overline{\mathbf{H}}_2 \end{pmatrix},
\end{aligned}$$

where $\bar{\ell} \coloneqq \max\{0, k+\ell-m\}$. The submatrices have dimensions $\overline{\mathbf{H}}_1 \in \mathbb{F}_z^{(n-m-\bar{\ell})\times(k+\ell)}$, $\overline{\mathbf{H}}_2 \in \mathbb{F}_z^{\bar{\ell}\times(k+\ell)}$, $\mathbf{H}_1 \in \mathbb{F}_p^{(n-k-\ell)\times(k+\ell)}$, and $\mathbf{H}_2 \in \mathbb{F}_p^{\ell\times(k+\ell)}$. We define the lists

$$\begin{aligned}
\mathcal{L}_a &\coloneqq \left\{ \left(\bar{\mathbf{e}}_a, \quad (\bar{\mathbf{e}}_a, \mathbf{0})\overline{\mathbf{H}}_2^\top, \qquad (g^{\bar{\mathbf{e}}_a}, \mathbf{0})\mathbf{H}_2^\top \right) \mid \bar{\mathbf{e}}_a \in \mathbb{F}_z^{\lfloor \frac{k+\ell}{2} \rfloor} \right\}, \text{ and} \\
\mathcal{L}_b &\coloneqq \left\{ \left(\bar{\mathbf{e}}_b, -(\mathbf{0}, \bar{\mathbf{e}}_b)\overline{\mathbf{H}}_2^\top, \mathbf{s}_2 - (\mathbf{0}, g^{\bar{\mathbf{e}}_b})\mathbf{H}_2^\top \right) \mid \bar{\mathbf{e}}_b \in \mathbb{F}_z^{\lceil \frac{k+\ell}{2} \rceil} \right\},
\end{aligned}$$

which contain $L_a \coloneqq |\mathcal{L}_a| = z^{\lfloor \frac{k+\ell}{2} \rfloor}$ and $L_b \coloneqq |\mathcal{L}_b| = z^{\lceil \frac{k+\ell}{2} \rceil}$ elements, respectively. Here, the second and the third components of each list element form the corresponding label. A concatenation $(\bar{e}_a, \bar{e}_b) \in \mathbb{F}_z^{k+\ell}$ results in a solution of the small instance if and only if both parts of the corresponding labels match:

$$\begin{aligned}
(\bar{\mathbf{e}}_a, \mathbf{0})\,\overline{\mathbf{H}}_2^\top &= \quad -(\mathbf{0}, \bar{\mathbf{e}}_b)\,\overline{\mathbf{H}}_2^\top & &\iff & (\bar{\mathbf{e}}_a, \bar{\mathbf{e}}_b)\overline{\mathbf{H}}_2^\top &= \mathbf{0}, \\
\left(g^{\bar{\mathbf{e}}_a}, \mathbf{0}\right)\mathbf{H}_2^\top &= \mathbf{s}_2 - \left(\mathbf{0}, g^{\bar{\mathbf{e}}_b}\right)\mathbf{H}_2^\top & &\iff & (g^{\bar{\mathbf{e}}_a}, g^{\bar{\mathbf{e}}_b})\mathbf{H}_2^\top &= \mathbf{s}_2.
\end{aligned}$$

Hence, solutions of the small instance can be found via a collision search performed on the label of the list elements. Since the labels live in $\mathbb{F}_z^{\bar{\ell}} \times \mathbb{F}_p^{\ell}$, this search yields on average

$$\frac{L_a \cdot L_b}{z^{\bar{\ell}} \cdot p^{\ell}} = z^{k+\ell-\bar{\ell}} p^{-\ell}$$

collisions, which are extended to the complete instance.

Theorem 13. The presented collision-based solver $\mathsf{COLL}(G)$ requires at least

$$M_{\mathsf{COLL}(G)}(p, n, k, z, m) = L_a \cdot \left\lfloor \frac{k+\ell}{2} \right\rfloor \cdot \log_2(z)$$

bits of memory. The number of binary operations can be bounded from below as

$$C_{\mathsf{COLL}(G)}(p, n, k, z, m) = \min_{\ell} \left\{ \frac{C_a + C_b + C_c}{1 + (z^m - 1)\,p^{k-n}} \log_2\!\left(M_{\mathsf{COLL}(G)}(p, n, k, z, m)\right) \right\},$$

where C_a, C_b and C_c are bounded as

$$\begin{aligned}
C_a &= L_a \cdot \left(\left\lfloor \frac{k+\ell}{2} \right\rfloor \cdot \log_2(z) + \bar{\ell} \cdot \log_2(z) + \ell \cdot \log_2(p) \right), \\
C_b &= L_b \cdot \left(\left\lceil \frac{k+\ell}{2} \right\rceil \cdot \log_2(z) + \bar{\ell} \cdot \log_2(z) + \ell \cdot \log_2(p) \right), \\
C_c &= L_a \cdot L_b \cdot z^{-\bar{\ell}} \cdot p^{-\ell} \cdot (k+\ell) \cdot \log_2(p),
\end{aligned}$$

with $\bar{\ell} = \max\{0, k+\ell-m\}$.

Proof. To perform the collision search, the algorithm has to store the smaller list among $\mathcal{L}_a$ and $\mathcal{L}_b$. Since this list contains dense vectors of length $\lfloor \frac{k+\ell}{2} \rfloor$ with entries in $\mathbb{E}$, at least $\lfloor \frac{k+\ell}{2} \rfloor \log_2(z)$ bits are required per list element. This gives the bound on the memory cost.

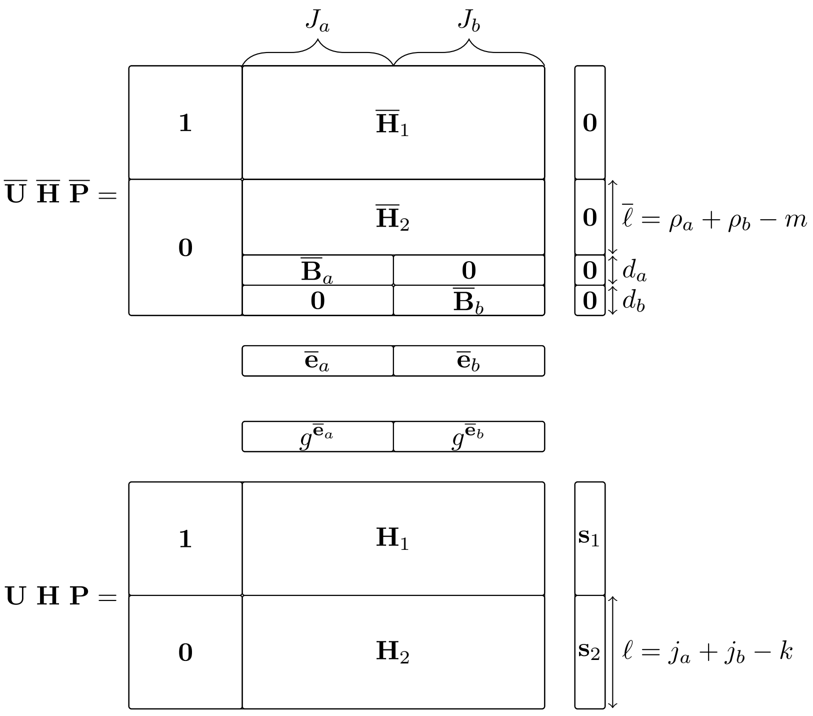
J_a
J_b
$\overline{\mathbf{U}}\,\overline{\mathbf{H}}\,\overline{\mathbf{P}} =$
$\mathbf{1}$
$\overline{\mathbf{H}}_1$
$\mathbf{0}$
$\overline{\mathbf{H}}_2$
$\overline{\mathbf{B}}_a$
$\overline{\mathbf{B}}_b$
$\overline{\ell} = \rho_a + \rho_b - m$
d_a
d_b
$\overline{\mathbf{e}}_a$
$\overline{\mathbf{e}}_b$
$g^{\overline{\mathbf{e}}_a}$
$g^{\overline{\mathbf{e}}_b}$
$\mathbf{U}\,\mathbf{H}\,\mathbf{P} =$
$\mathbf{H}_1$
$\mathbf{H}_2$
$\mathbf{s}_1$
$\mathbf{s}_2$
$\ell = j_a + j_b - k$

Figure 2: Illustration of the collision-based solver with subgroup G and small-support subcodes.

Let us consider the solver's time complexity, estimated as the cost of finding a particular solution divided by the number of solutions. According to Theorem 5, the average number of solutions is given by $1 + (z^m - 1)p^{k-n}$.

First, one computes two syndromes for the error vectors $\overline{\mathbf{e}}_a$ associated with list $\mathcal{L}_a$. Each error vector has a size of $\lfloor \frac{k+\ell}{2} \rfloor \cdot \log_2(z)$ bit. The syndromes have sizes $\ell \cdot \log_2(p)$ bit and $\overline{\ell} \cdot \log_2(z)$ bit. Hence, we estimate the required number of binary operations as $L_a \cdot \big(\lfloor \frac{k+\ell}{2} \rfloor \cdot \log_2(z) + \overline{\ell} \cdot \log_2(z) + \ell \cdot \log_2(p)\big)$.

Next, two syndromes are computed for each error vector $\overline{\mathbf{e}}_b$ associated with list $\mathcal{L}_b$. Again, due to the size of the objects, this requires at least $L_b \cdot \big(\lceil \frac{k+\ell}{2} \rceil \cdot \log_2(z) + \overline{\ell} \cdot \log_2(z) + \ell \cdot \log_2(p)\big)$ binary operations.

The collision search yields on average $L_a \cdot L_b \cdot z^{-\overline{\ell}} \cdot p^{-\ell}$ partial solutions. For each collision, one checks whether $\mathbf{e}_2$ extends to a solution of the complete problem. Even for false positives, one has to calculate at least one syndrome symbol of the complete instance, which is the sum of $k + \ell$ elements of $\mathbb{F}_p$. Hence, this step requires at least $L_a \cdot L_b \cdot p^{-\ell} \cdot (k + \ell) \cdot \log_2(p)$ binary operations.

Finally, the memory access cost is modeled with the conservative logarithmic cost model [2, 12]. That is, the cost per iteration is increased by a factor $\log_2(M_{\mathsf{COLL}(G)}(p, n, k, z))$. □

Collision search with subgroup G and small-support subcodes: While $\mathsf{COLL}(G)$ uses a random permutation, the solver's computational cost can be reduced by choosing the permutation based on the structure of G. This improvement is illustrated in Figure 2.

The attacker begins by searching for a public key with a subgroup G, for which $\langle \mathbf{H} \rangle$ contains two subcodes with the following properties:

- The first subcode is of dimension d_a and has a support J_a of size $j_a = |J_a|$. Hence, this first subcode is generated by $(\mathbf{0}\,\overline{\mathbf{B}}_a\,\mathbf{0})$ with $\overline{\mathbf{B}}_a \in \mathbb{F}_z^{d_a \times j_a}$.
- The second subcode is of dimension d_b and has a support J_b of size $j_b = |J_b|$. Hence, the second subcode is generated by $(\mathbf{0}\,\overline{\mathbf{B}}_b\,\mathbf{0})$ with $\overline{\mathbf{B}}_b \in \mathbb{F}_z^{d_b \times j_b}$.
- The supports J_a and J_b are disjoint.

Due to the obvious connection to the codeword finding problem, the decisional version of this problem can be shown to be NP-complete itself [3, Theorem 2]. The best-known approach to solving the computational version is given by information set decoding. Any solver can only succeed if such subcodes indeed exist for a given G. In accordance with [25, Theorem 1], we estimate that a subcode of dimension d and support size j exists with probability

$$P(j,d) = \min\left\{ \binom{n}{j} (z^d-1)^{j-d} \begin{bmatrix} n-m \\ d \end{bmatrix}_z \begin{bmatrix} n \\ d \end{bmatrix}_z^{-1}, 1 \right\},$$

where $\begin{bmatrix} n \\ d \end{bmatrix}_z$ denotes the Gaussian binomial, computed as

$$\begin{bmatrix} n \\ d \end{bmatrix}_z = \prod_{i=1}^{d} \frac{q^{n-i+1}-1}{q^i-1}.$$

Hence, the probability that both subcodes exist in $\langle \overline{\mathbf{H}} \rangle$ can be upper-bounded as $P(j_a,d_a)\cdot P(j_b,d_b)$. It follows that $(P(j_a,d_a)\cdot P(j_b,d_b))^{-1}$ many restrictions G have to be considered on average to find one which allows for subcodes with parameters j_a, d_a, j_b, d_b.

Example 14. For the parameters $p = 509$, $z = 127$, $m = 25$ and $n = 55$, $k = 36$, a one-dimensional subcode with support size 19 exists with probability 0.45. Further, a four-dimensional subcode with support size 23 exists with probability 2^{-116}.

One begins with bringing $\mathbf{H}$ and $\overline{\mathbf{H}}$ into quasi-systematic form with respect to the subcodes. That is, the column permutation is chosen such that the support of the subcodes $J_a \cup J_b$ is permuted to the last $j_a + j_b$ positions. Then, partial Gaussian elimination yields parity-check matrices of the form given in Figure 2.

The submatrices have dimensions $\overline{\mathbf{H}}_1 \in \mathbb{F}_z^{(n-m-\overline{\ell})\times(j_a+j_b)}$, $\overline{\mathbf{H}}_2 \in \mathbb{F}_z^{\overline{\ell}\times(j_a+j_b)}$, $\mathbf{H}_1 \in \mathbb{F}_p^{(n-k-\ell)\times(j_a+j_b)}$, and $\mathbf{H}_2 \in \mathbb{F}_p^{\ell\times(j_a+j_b)}$ where $\ell = j_a + j_b - k$ and $\overline{\ell} := \max\{0, j_a+j_b-d_a-d_b-m\}$. We define $\rho_a := j_a - d_a$, $\rho_b := j_b - d_b$ and write $\overline{\ell} := \max\{0, \rho_a+\rho_b-m\}$. One constructs the lists

$$\mathcal{L}_a := \left\{ \left(\overline{\mathbf{e}}_a, \quad (\overline{\mathbf{e}}_a, \mathbf{0})\overline{\mathbf{H}}_2^\top, \qquad \left(g^{\overline{\mathbf{e}}_a}, \mathbf{0}\right)\mathbf{H}_2^\top \right) \mid \overline{\mathbf{e}}_a \in \ker(\overline{\mathbf{B}}_a) \right\},$$
$$\mathcal{L}_b := \left\{ \left(\overline{\mathbf{e}}_b, -(\mathbf{0}, \overline{\mathbf{e}}_b)\overline{\mathbf{H}}_2^\top, \mathbf{s}_2 - \left(\mathbf{0}, g^{\overline{\mathbf{e}}_b}\right)\mathbf{H}_2^\top \right) \mid \overline{\mathbf{e}}_b \in \ker(\overline{\mathbf{B}}_b) \right\}.$$

These lists contain $L_a = |\mathcal{L}_a| = z^{j_a-d_a} = z^{\rho_a}$ and $L_b = |\mathcal{L}_b| = z^{j_b-d_b} = z^{\rho_b}$ elements. Thus, the list sizes are smaller compared to the basic collision-based solver. This also impacts the collision search, which yields on average

$$\frac{L_a \cdot L_b}{z^{\max(\rho_a+\rho_b-m,0)} \cdot p^{j_a+j_b-k}} = \frac{z^{\rho_a}\cdot z^{\rho_b}}{z^{\overline{\ell}}\cdot p^{\ell}}$$

partial solutions, for which one then checks whether they can be extended to a solution of the original problem.

Theorem 15. The collision-based solver with small-support subcodes $\mathsf{SUBC}(G)$ requires at least

$$M_{\mathsf{SUBC}(G)}(p,n,k,z,m) = \min\{L_a\cdot\rho_a, L_b\cdot\rho_b\}\cdot\log_2(z),$$

bits of memory, where $L_a = z^{\rho_a}$ and $L_b = z^{\rho_b}$. At least

$$C_{\mathsf{SUBC}(G)}(p,n,k,z,m) = \min_{J_a,J_b}\left\{ \frac{C_a+C_b+C_c}{1+(z^m-1)p^{k-n}} \log_2\left(M_{\mathsf{SUBC}(G)}(p,n,k,z,m)\right) + \frac{1}{P(j_a,d_a)\cdot P(j_b,d_b)} \right\}$$

binary operations are required. C_a, C_b and C_c are given by

$$C_a = L_a\cdot\left(\rho_a\cdot\log_2(z) + \overline{\ell}\cdot\log_2(z) + \ell\cdot\log_2(p)\right),$$
$$C_b = L_b\cdot\left(\rho_b\cdot\log_2(z) + \overline{\ell}\cdot\log_2(z) + \ell\cdot\log_2(p)\right),$$
$$C_c = L_a\cdot L_b\cdot z^{-\overline{\ell}}\cdot p^{-\ell}(j_a+j_b)\log_2(p),$$

where $\ell = j_a + j_b - k$ and $\overline{\ell} = \max\{0, \rho_a+\rho_b-m\}$.

Table 3: Bit-complexity estimates for solvers of R-SDP(G) with parameters as used by CROSS. The restriction is given by $\mathbb{E} = \{16^i \mid i \in \mathbb{F}_{127}\} \subset \mathbb{F}_{509}$.

Category	Parameters			Solver	Solver parameters						Cost [bit]	
	n	k	m		ℓ	$\overline{\ell}$	j_a	j_b	ρ_a	ρ_b	Time	Mem.
NIST 1	55	36	25	COLL(G)	4		20	20			152	146
				SUBC(G)	6	12	19	23	18	19	143	132
				[9]							145	64
NIST 3	79	48	40	COLL(G)	9		28	29			217	203
				SUBC(G)	10	15	28	30	27	28	210	196
				[9]							212	96
NIST 5	106	69	48	COLL(G)	8		38	39			286	273
				SUBC(G)	9	25	37	41	36	37	272	259
				[9]							276	128

Proof. The cost is obtained in a similar way as in Theorem 13.

To perform the collision search, the smaller list among $\mathcal{L}_a$ and $\mathcal{L}_b$ is stored. Without loss of generality, we assume in the following that this is $\mathcal{L}_a$. Since this list contains elements of $\ker(\overline{\mathbf{B}}_a)$, at least $\rho_a \log_2(z)$ bits are required per list element, resulting in the bound on the memory cost $M_{\mathsf{SUBC}(G)}(p, n, k, z, m)$.

Let us consider the algorithm's time complexity $C_{\mathsf{SUBC}(G)}(p, n, k, z, m)$, which is estimated as the cost of finding a particular solution divided by the number of solutions. According to Theorem 5, the average number of solutions is given by $1 + (z^m - 1)p^{k-n}$.

First, one computes two syndromes for the error vectors $\overline{\mathbf{e}}_a$ associated with list $\mathcal{L}_a$. Each error vector can be represented using $\rho_a \cdot \log_2(z)$ bits. The label has a size of $\overline{\ell} \cdot \log_2(z) + \ell \cdot \log_2(p)$ bits with $\ell = j_a + j_b - k$ and $\overline{\ell} = \max\{0, \rho_a + \rho_b - m\}$. Hence, we estimate the required number of binary operations as $L_a \cdot \big(\rho_a \cdot \log_2(z) + \overline{\ell} \cdot \log_2(z) + \ell \cdot \log_2(p)\big)$ binary operations.

Next, two syndromes are computed for each error vector $\overline{\mathbf{e}}_b$ associated with list $\mathcal{L}_b$. Again, due to the size of the objects, this requires at least $L_a \big(\rho_b \cdot \log_2(z) + \overline{\ell} \cdot \log_2(z) + \ell \cdot \log_2(p)\big)$ binary operations.

The collision search yields on average $L_a \cdot L_b \cdot z^{-\overline{\ell}} \cdot p^{-\ell}$ partial solutions. For each collision, one checks whether $\mathbf{e}_2$ extends to a solution of the original problem. Even for false positives, one has to calculate at least one syndrome symbol of the original instance, which is the sum of $j_a + j_b$ elements of $\mathbb{F}_p$. Hence, this step requires at least $L_a \cdot L_b \cdot z^{-\overline{\ell}} \cdot p^{-\ell} \cdot (j_a + j_b) \cdot \log_2(p)$ binary operations.

Finally, the memory access cost is modeled via the conservative logarithmic cost model [12, 2]. That is, the cost per iteration is increased by a factor $\log_2(M_{\mathsf{SUBC}(G)}(p, n, k, z, m))$. □

A comment on representation-based solvers: Representation-based solvers cannot utilize the structure of G in the presented way since they rely on sum partitions. Unlike concatenation, summation does not align with the structure of the exponentiation.

Expected security strength: Table 3 presents the computational cost of the presented solvers. The optimized solver parameters are given and can be reproduced using the code available at `https://www.cross-crypto.com/resources`. The combination of collision search and small-support subcodes yields the best performance for each of the parameter sets, which achieve NIST security categories 1, 3, and 5. Note that the costs of the solver of [9] are given in $\mathbb{F}_p$ operations.

4 Proof of Security

The security proof of CROSS is split into two sections: in Section 4.2, we analyze the security of the Zero-Knowledge ZK protocol 4 and in Section 4.4 the security of the signature scheme after the Fiat-Shamir transform.

4.1 Basics on ZK protocols

We first recall the main definitions for ZK protocols.

Definition 16 (Interactive Proof)**.** An *interactive proof* $\Pi = (\mathcal{P}, \mathcal{V})$ for a binary relation $R \subseteq X \times Y$ is an interactive protocol between two probabilistic polynomial-time machines $\mathcal{P}$ and $\mathcal{V}$.

The *prover* $\mathcal{P}$ takes as input a pair $(x, y) \in R$ while the *verifier* $\mathcal{V}$ takes as input x. As the output of the protocol - denoted by $(\mathcal{P}(y), \mathcal{V})(x)$ - $\mathcal{V}$ either accepts (outputs 1) or rejects (outputs 0).

We say that a *transcript*, i.e., the set of all messages exchanged in a protocol execution, is *accepting* (*rejecting*) if $\mathcal{V}$ accepts (rejects, respectively).

Throughout this security guide, we assume that, within an execution of an interactive proof $(\mathcal{P}, \mathcal{V})$, the prover $\mathcal{P}$ always sends the first and the last message. Hence, the number of communication rounds is odd, i.e., of the form $2\mu + 1$ with μ a positive integer.

We refer to an interactive proof having $2\mu + 1$ communication rounds with the name $(2\mu + 1)$-*round interactive proof*. In the case of CROSS, we have that $\mu = 2$; however, since the results presented here hold for any μ, we describe them in full generality.

Definition 17 (Public-Coin)**.** An interactive proof $\Pi = (\mathcal{P}, \mathcal{V})$ is *public-coin* if all $\mathcal{V}$'s random choices are made public.

If an interactive proof is public-coin, the verifier needs to send to the prover only their random choices. For this reason, we call the messages sent by the verifier *challenges* and refer to the set from which the verifier's messages are sampled as the *challenge set*.

In the case of a $(2\mu+1)-$round interactive proof, we define the challenge set $\texttt{Ch}$ as the Cartesian product of μ *round challenge sets* $\texttt{Ch}[i]$, with $i \in \{1, \dots, \mu\}$, meaning that the challenge for the i-th round is sampled from $\texttt{Ch}[i]$.

Definition 18 (Completeness)**.** An interactive proof $\Pi = (\mathcal{P}, \mathcal{V})$ for a binary relation $R \subseteq X \times Y$ is *complete* if, for every $(x, y) \in R$, we have:

$$\mathbb{P}[(\mathcal{P}(y), \mathcal{V})(x) = 0] \leq \rho(x),$$

where the *completeness error* $\rho(x)$ is negligible (in $|x|$). If $\rho(x) = 0$ for all $x \in L_R$, the protocol is said to be *perfectly complete*.

Definition 19 (Honest-Verifier Zero-Knowledge)**.** Let $\Pi = (\mathcal{P}, \mathcal{V})$ be an interactive proof system for an hard relation $R \subseteq X \times Y$. We say that Π is *(weak) computationally honest-verifier zero-knowledge* if there exists a probabilistic polynomial time algorithm $\mathcal{S}$, called the *simulator*, such that the following two distribution ensembles are computationally indistinguishable:

$$\{(x, \mathsf{transcript}(\mathcal{P}(x, y), \mathcal{V}(x))) \mid (x, y) \xleftarrow{\$} R\} \quad \text{and} \quad \{(x, \mathcal{S}(x)) \mid (x, y) \xleftarrow{\$} R\},$$

where $\mathsf{transcript}(\mathcal{P}(x, y), \mathcal{V}(x))$ denotes a transcript of an honest execution between a prover, knowing both x and y, and a verifier, knowing only x.

Definition 20 (Knowledge Soundness)**.** An interactive proof $(\mathcal{P}, \mathcal{V})$ for a binary relation $R \subseteq X \times Y$ is *knowledge-sound*, with *knowledge error* κ, if there exists an algorithm $\mathcal{E}$ that, given as input any $x \in X$ and rewindable oracle access to a (potentially-dishonest) prover $\mathcal{P}^*$, runs in an expected polynomial time (in $|x|$) and outputs a witness $y \in Y$ for x with probability:

$$\mathbb{P}[(x, \mathcal{E}^{\mathcal{P}^*}(x)) \in R] \geq \frac{\varepsilon(x, \mathcal{P}^*) - \kappa(x)}{\texttt{poly}(|x|)},$$

where $\varepsilon(x, \mathcal{P}^*) = \mathbb{P}[(\mathcal{P}^*, \mathcal{V})(x) = 1]$. The algorithm $\mathcal{E}$ is called *knowledge extractor*.

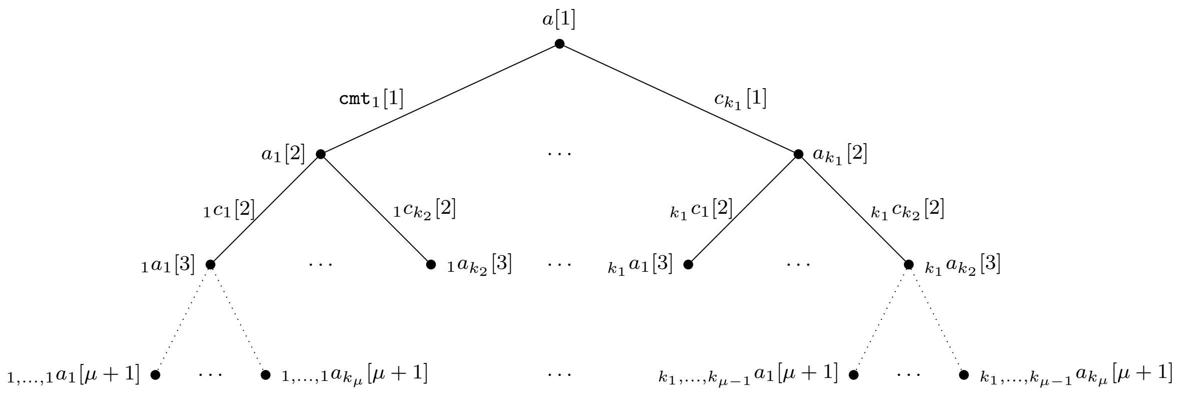

Figure 3: Graphical representation of a $(k_1, \dots, k_\mu)-$tree of transcripts for a $(2\mu+1)-$round public-coin interactive proof. Left subscripts represent the ancestor nodes, superscripts represent the corresponding round, while right subscripts are used to enumerate edges originating from a node and their corresponding arrival nodes.

Definition 21 (Tree of Transcripts)**.** Let $k_1, \dots, k_\mu, N_1, \dots, N_\mu$ be positive integers, $R \subseteq X \times Y$ be a binary relation and $\Pi = (\mathcal{P}, \mathcal{V})$ a $(2\mu+1)-$round public-coin interactive proof for R, where $\mathcal{V}$ samples $i-$th challenges ($i \in \{1, \dots, \mu\}$) from a set $\mathtt{Ch}[i]$ of cardinality $N_i \geq k_i$.

A $(k_1, \dots, k_\mu)$*-tree of transcripts* for $(\mathcal{P}, \mathcal{V})$ is a set of $K = \prod_{i=1}^{\mu} k_i$ transcripts relative to a given statement $x \in X$, arranged in the following tree structure, where nodes correspond to prover's messages while edges to verifier's challenges.

From every node at level i, with $i \in \{1, \dots, \mu\}$, exactly k_i edges originate, corresponding to k_i pairwise-distinct challenges belonging to $\mathtt{Ch}[i]$. Then, each of the K transcripts corresponds to exactly one path from the root node to a leaf node.

A graphical representation of a tree of transcripts is provided in Figure 3, where $a[1]$ denotes the prover's first message, $\mathtt{cmt}_1[1], \dots, \mathtt{cmt}_{k_1}[1]$ are sampled from $\mathtt{Ch}[1]$, and so on.

Definition 22 ($(k_1, \dots, k_\mu)$-Special Soundness)**.** Let $k_1, \dots, k_\mu$, $N_1, \dots, N_\mu$ be positive integers and $R \subseteq X \times Y$ be a binary relation. A $(2\mu+1)$-round public-coin interactive proof $(\mathcal{P}, \mathcal{V})$ for R, where $\mathcal{V}$ samples $i-$th challenges ($i \in \{1, \dots, \mu\}$) from a set $\mathtt{Ch}[i]$ of cardinality $N_i \geq k_i$, is $(k_1, \dots, k_\mu)$*-out-of-*$(N_1, \dots, N_\mu)$ *special sound*, or simply $(k_1, \dots, k_\mu)$*-special sound*, if there exists a polynomial-time algorithm that, on input a true statement $x \in X$ and a $(k_1, \dots, k_\mu)$-tree of accepting transcripts for $(\mathcal{P}, \mathcal{V})$ and relative to x, outputs a witness $y \in Y$ for x.

4.2 Security of the Protocol

Let us quickly recall the ZK protocol CROSS-ID in Figure 4. The protocol as well as the proofs use the subgroup G, which also includes the R-SDP case, by setting $G = \mathbb{E}^n$.

Proposition 23 (Completeness)**.** The CROSS-ID protocol in Figure 4 is complete.

Proof. We have to show that the honest prover always gets accepted. When $\mathtt{chall}_2 = 0$, we have

$$\mathbf{y}' = \mathbf{v} \star \mathbf{y} = \mathbf{v} \star \mathbf{u}' + \mathtt{chall}_1 \mathbf{v} \star \mathbf{e}' = \mathbf{u} + \mathtt{chall}_1 \mathbf{e}.$$

So, it holds that

$$\mathbf{y}'\mathbf{H}^\top - \mathtt{chall}_1 \mathbf{s} = \mathbf{u}\mathbf{H}^\top + \mathtt{chall}_1 \mathbf{e}\mathbf{H}^\top - \mathtt{chall}_1 \mathbf{s} = \mathbf{u}\mathbf{H}^\top + \mathtt{chall}_1 \mathbf{s} - \mathtt{chall}_1 \mathbf{s} = \mathbf{u}\mathbf{H}^\top = \mathbf{s}'.$$

This indeed corresponds to the syndrome that was used to generate $\mathtt{cmt}_0$. Finally, we also verify that $\mathbf{v} \in G$. When $\mathtt{chall}_2 = 1$, the prover provides only seeds: since PRNGs are deterministic, the verifier obtains the very same quantities that have been used to generate the commitment $\mathtt{cmt}_1$. $\square$

```
Private Key $\mathbf{e} \in G$
Public Key  $G \subseteq \mathbb{E}^n$, $\mathbf{H} \in \mathbb{F}_p^{(n-k)\times n}$, $\mathbf{s} = \mathbf{e}\mathbf{H}^\top \in \mathbb{F}_p^{n-k}$

PROVER                                                              VERIFIER

// Sampling Seed to compute $\mathbf{e}', \mathbf{u}'$
Seed $\xleftarrow{\$} \{0,1\}^\lambda$
$(\mathbf{e}', \mathbf{u}') \leftarrow$ CSPRNG(Seed) // with co − domain $G \times \mathbb{F}_p^n$

// Computing $\mathbf{v}, \mathbf{u}, \mathbf{s}'$
$\mathbf{v} \leftarrow \mathbf{e} \star (\mathbf{e}')^{-1}$
$\mathbf{u} \leftarrow \mathbf{v} \star \mathbf{u}'$
$\mathbf{s}' \leftarrow \mathbf{u}\mathbf{H}^\top$

// Computing commitments
$\mathtt{cmt}_0 \leftarrow \mathsf{Hash}(\mathbf{s}'|\mathbf{v})$
$\mathtt{cmt}_1 \leftarrow \mathsf{Hash}(\mathbf{u}'|\mathbf{e}')$
                                   $\xrightarrow{\mathtt{cmt}_0,\mathtt{cmt}_1}$
                                            // Sampling first challenge
                                   $\xleftarrow{\mathtt{chall}_1}$
                                                              $\mathtt{chall}_1 \xleftarrow{\$} \mathbb{F}_p^*$
// Computing first response
$\mathbf{y} \leftarrow \mathbf{u}' + \mathtt{chall}_1\mathbf{e}'$
$\mathtt{digest}_\mathbf{y} \leftarrow \mathsf{Hash}(\mathbf{y})$
                                   $\xrightarrow{\mathtt{digest}_\mathbf{y}}$
                                            // Sampling second challenge
                                                              $\mathtt{chall}_2 \xleftarrow{\$} \{0,1\}$
                                   $\xleftarrow{\mathtt{chall}_2}$
// Computing second response
If $\mathtt{chall}_2 = 0$, resp $\leftarrow (\mathbf{y}, \mathbf{v})$
If $\mathtt{chall}_2 = 1$, resp $\leftarrow$ Seed
                                   $\xrightarrow{\mathtt{resp}}$
                                            // Verification
                                            If $\mathtt{chall}_2 = 0$:
                                              $\mathbf{y}' \leftarrow \mathbf{v} \star \mathbf{y}$
                                              $\mathbf{s}' \leftarrow \mathbf{y}'\mathbf{H}^\top - \mathtt{chall}_1\mathbf{s}$
                                              Accept if:
                                                1) $\mathsf{Hash}(\mathbf{y}) = \mathtt{digest}_\mathbf{y}$
                                                2) $\mathsf{Hash}(\mathbf{s}'|\mathbf{v}) = \mathtt{cmt}_0$
                                                3) $\mathbf{v} \in G$
                                            If $\mathtt{chall}_2 = 1$:
                                              $(\mathbf{e}', \mathbf{u}') \leftarrow$ CSPRNG(Seed) // with co − domain $G \times \mathbb{F}_p^n$
                                              $\mathbf{y} \leftarrow \mathbf{u}' + \mathtt{chall}_1\mathbf{e}'$
                                              Accept if:
                                                1) $\mathsf{Hash}(\mathbf{y}) = \mathtt{digest}_\mathbf{y}$
                                                2) $\mathsf{Hash}(\mathbf{u}'|\mathbf{e}') = \mathtt{cmt}_1$
```

Figure 4: CROSS-ID

Proposition 24 (Honest-Verifier Zero-Knowledge)**.** The CROSS-ID protocol in Figure 4 is (weak) honest-verifier zero-knowledge in the ROM.

Proof. We prove that a simulator $\mathcal{S}$ with knowledge of the challenges can easily simulate the interaction $(\mathcal{P}, \mathcal{V})$ between the prover and the verifier. Formally, we show that $\mathcal{S}$ produces a transcript T^* that is indistinguishable from the transcript T resulting from the interaction between $\mathcal{P}$ and $\mathcal{V}$. $\mathcal{S}$ starts sampling a random bit $\mathtt{chall}_2$. Then, depending of the value of $\mathtt{chall}_2$, $\mathcal{S}$ does the following:

- $\mathtt{chall}_2 = 0$: the simulator picks a random $\mathtt{chall}_1 \in \mathbb{F}_p^*$, then computes $\mathbf{e}^* \in \mathbb{F}_p^n$ such that $\mathbf{e}^*\mathbf{H}^\top = \mathbf{s}$. Then, $\mathcal{S}$ selects a random $\mathbf{v}^* \in G$ and a vector $\mathbf{u}^* \in \mathbb{F}_p^n$, and computes $\mathbf{u}'^* = \mathbf{v}^{*-1} \star \mathbf{e}^*$.

  Finally, it computes $\mathbf{s}^{*\prime} = \mathbf{u}^*\mathbf{H}^\top$ and $\mathtt{cmt}_0 = \mathsf{Hash}(\mathbf{s}^{*\prime}, \mathbf{v}^*)$. Then, $\mathcal{S}$ computes $\mathbf{y}^* = \mathbf{u}'^* + \mathtt{chall}_1\mathbf{e}'^*$. Finally, $\mathcal{S}$ set $\mathtt{cmt}_1$ as a random binary string with length 2λ. Since $\mathtt{chall}_2 = 0$ this commitment is never revealed, and thus, in the ROM, this is computationally indistinguishable from an honestly computed $\texttt{cmt}_1$.

It is easy to see that the transcript produced by $\mathcal{S}$ (i.e., the values $\mathbf{y}^*$ and $\mathbf{v}^*$) follows the same statistical distribution as those of an honestly produced transcript.

Indeed, in an honest execution, $\mathbf{y}$ is uniformly random over $\mathbb{F}_p^*$ because $\mathbf{u}'$ is uniformly random over $\mathbb{F}_p^n$. This guarantees that $\mathbf{u}' + \texttt{chall}_1\mathbf{e}'$ is uniformly random over $\mathbb{F}_p^n$, and the same holds after multiplying with $\mathbf{v}$.

Finally, in an honest execution of the protocol, $\mathbf{v}$ is uniformly distributed over G. Indeed, for any $\mathbf{e}' \in G$ there is a unique $\mathbf{v} \in G$ such that $\mathbf{v} \star \mathbf{e}' = \mathbf{e}$. If $\mathbf{e}'$ is uniformly random over G, then $\mathbf{v}$ also follows the same distribution.

- $\texttt{chall}_2 = 1$: in this case, the simulator simply executes the protocol by sampling the seed and computing $\texttt{cmt}_1$ analogously to what the honest prover $\mathcal{P}$ would do. For the other commitment, $\texttt{cmt}_0$, it is enough to use a random binary string again.

□

Remark 25. Notice that, as pointed out in [20], CROSS-ID is only computationally honest-verifier zero-knowledge, since λ-bit seeds are used in the real transcript and the number of possible values of $\texttt{cmt}_0$ and $\texttt{cmt}_1$ are reduced. Moreover, since there is no randomness involved in the computation of the commitments, CROSS-ID is only *weak* honest-verifier zero-knowledge. As explained in Section 4.3, this is not a problem for the EUF-CMA security of CROSS, however this may not be sufficient when considering advanced functionalities (e.g. ring signature), where *strong* honest-verifier zero-knowledge (i.e., the ensembles of Definition 19 are indistinguishable also when the witness y is revealed) is required.

To solve this issue, it is possible to include a nonce in the computation of $\texttt{cmt}_0$ and $\texttt{cmt}_1$, however doing so would lead worse performances as it would increase the signature size.

Proposition 26 (Soundness)**.** The protocol in Figure 4 is $(2, 2)$-special sound.

Proof. We consider four accepting transcripts T_1, T_2, T_3, T_4, all associated with the same pair of commitments $\texttt{cmt}_0, \texttt{cmt}_1$. The commitment $\texttt{cmt}_0$ fixes the pair $(\mathbf{s}', \mathbf{v})$, while the commitment $\texttt{cmt}_1$ fixes the pair $(\mathbf{u}', \mathbf{e}')$.

We identify the transcripts by the challenge values, which we denote respectively by $(\texttt{chall}_1, 0)$, $(\texttt{chall}_1, 1)$, $(\texttt{chall}_1^*, 0)$, and $(\texttt{chall}_1^*, 1)$. Taking into account the prover's replies, we have that the transcripts are structured as follows:

T_1: $\big(\texttt{cmt}_0, \texttt{cmt}_1, \texttt{chall}_1, \texttt{digest}_\mathbf{y}, \mathbf{y}, \mathbf{v}\big)$;

T_2: $\big(\texttt{cmt}_0, \texttt{cmt}_1, \texttt{chall}_1, \texttt{digest}_\mathbf{y}, \texttt{Seed}\big)$;

T_3: $\big(\texttt{cmt}_0, \texttt{cmt}_1, \texttt{chall}_1^*, \texttt{digest}_\mathbf{y}^*, \mathbf{y}^*, \mathbf{v}^*\big)$;

T_4: $\big(\texttt{cmt}_0, \texttt{cmt}_1, \texttt{chall}_1^*, \texttt{digest}_\mathbf{y}^*, \texttt{Seed}^*\big)$.

We now show that, from the knowledge of these four transcripts, a solution for the R-SDP(G) instance $\{\mathbf{s}, \mathbf{H}\}$ can be easily computed (i.e., in polynomial time).

We first focus on T_2 and T_4. Let $\mathbf{u}', \mathbf{e}'$ be the vectors generated from $\texttt{Seed}$, and $\mathbf{u}'^*, \mathbf{e}'^*$ those generated from $\texttt{Seed}^*$. Since $\texttt{cmt}_1$ is verified in both cases, either hash collisions have been found (i.e., $\mathsf{Hash}\big(\mathbf{u}', \mathbf{e}'\big) = \mathsf{Hash}\big(\mathbf{u}'^*, \mathbf{e}'^*\big)$ but $\mathbf{u}' \neq \mathbf{u}'^*$ and/or $\mathbf{e}' \neq \mathbf{e}'^*$), or $\mathbf{u}' = \mathbf{u}'^*$ and $\mathbf{e}' = \mathbf{e}'^*$.

Since also $\texttt{digest}_\mathbf{y}$ and $\texttt{digest}_\mathbf{y}^*$ are checked, and unless hash collisions have been found, this guarantees that $\texttt{digest}_\mathbf{y} = \mathsf{Hash}(\mathbf{y})$, where $\mathbf{y} = \mathbf{u}' + \texttt{chall}_1\mathbf{e}'$, and $\texttt{digest}_\mathbf{y}^* = \mathsf{Hash}(\mathbf{y}^*)$, where $\mathbf{y}^* = \mathbf{u}'^* + \texttt{chall}_1^*\mathbf{e}'^* = \mathbf{u}' + \texttt{chall}_1^*\mathbf{e}'$.

This implies that $\mathbf{y} - \mathbf{y}^* = \mathbf{e}'(\texttt{chall}_1 - \texttt{chall}_1^*)$.

Now, we look at the pair of transcripts T_1 and T_3. Unless hash collisions have been found, we have $\mathbf{v} = \mathbf{v}^*$,

$$(\mathbf{v} \star \mathbf{y})\mathbf{H}^\top - \texttt{chall}_1\mathbf{s} = \mathbf{s}', \text{ and } (\mathbf{v} \star \mathbf{y}^*)\mathbf{H}^\top - \texttt{chall}_1^*\mathbf{s} = \mathbf{s}',$$

from which it follows that

$$\mathbf{v} \star \mathbf{y} - \mathbf{y}^* \mathbf{H}^\top = (\mathtt{chall}_1 - \mathtt{chall}_1^*)\mathbf{s}.$$

Exploiting the relations we derived from the pair (T_2, T_4), we obtain $\mathbf{y} - \mathbf{y}^* = (\mathtt{chall}_1 - \mathtt{chall}_1^*)\mathbf{e}'$, where $\mathbf{e}'$ is a restricted vector, hence

$$(\mathtt{chall}_1 - \mathtt{chall}_1^*)(\mathbf{v} \star \mathbf{e}')\mathbf{H}^\top = (\mathtt{chall}_1 - \mathtt{chall}_1^* \mathbf{s} \implies (\mathbf{v} \star \mathbf{e}')\mathbf{H}^\top = \mathbf{s}.$$

Since $\mathbf{v}, \mathbf{e}'$ have been verified, thus $\mathbf{v}, \mathbf{e}' \in G$; then $\mathbf{v} \star \mathbf{e}' \in G$ means that $\mathbf{v} \star \mathbf{e}'$ solves R-SDP(G) for the instance $\{\mathbf{H}, \mathbf{s}\}$. □

4.3 Signature Scheme Security in General

From [4], we have that any fixed-weight repetition of a special sound protocol is knowledge-sound. In particular, we obtain the following theorem:

Theorem 27 (Fixed-Weight Repetition of a $(k_1, \ldots, k_\mu)$-Special-Sound Multi-Round Interactive Proof)**.** Let $(\mathcal{P}, \mathcal{V})$ be a $(k_1, \ldots, k_\mu)$-special-sound $(2\mu+1)$-round interactive proof and $(\mathcal{P}^{t,w}, \mathcal{V}^{t,w})$ be the (t, w)-fixed-weight repetition of $(\mathcal{P}, \mathcal{V})$, where w, t are positive integers with $1 \le w \le t$. Then $(\mathcal{P}^{t,w}, \mathcal{V}^{t,w})$ is knowledge-sound with knowledge error $\kappa_{t,w}$, where $\kappa_{t,w}$ is the maximum, taken over $\alpha \in \{0, \ldots, t\}$, of the expression

$$\frac{\sum_{\ell=\max(0,w-t+\alpha)}^{\min(w,\alpha)} \binom{\alpha}{\ell}\binom{t-\alpha}{w-\ell} Z_0^\ell \,(Z_1 - Z_0)^{\alpha-\ell}\,(Z_2)^{w-\ell}(Z_1 - Z_2)^{t-\alpha-w+\ell}}{\binom{t}{w}(N_\mu - 1)^{t-w}(\prod_{i=1}^{\mu-1} N_i)^t},$$

where Z_0, Z_1, Z_2 are defined as follows:

$$Z_0 := \prod_{\ell=1}^{\mu-1} N_\ell,$$
$$Z_1 := \sum_{\ell=1}^{\mu} \left(\prod_{j=\ell+1}^{\mu} N_j\right)(k_\ell - 1)\left(\prod_{j=1}^{\ell-1}(N_j - k_j + 1)\right),$$
$$Z_2 := \sum_{\ell=1}^{\mu-1} \left(\prod_{j=\ell+1}^{\mu-1} N_j\right)(k_\ell - 1)\left(\prod_{j=1}^{\ell-1}(N_j - k_j + 1)\right).$$

As an immediate corollary we have that CROSS is knowledge-sound.

To conclude our analysis and prove the EUF-CMA security, we need the results of [5].

Definition 28 (Security against Impersonation under Passive Attack)**.** Let $R \subseteq X \times Y$ be a binary relation, Π a $2\mu + 1$-round proof system for R and λ the security parameter. Let V the verification function of Π, $a[i]$ the message that the prover sends during the $i-$th round and $\mathsf{Ch}[i]$ the challenge space of the $i-$th round. The impersonation experiment between a challenger and an impersonator $\mathcal{I}$ is defined as follows:

Algorithm 1: Impersonation experiment $\mathsf{Exp}^{\mathcal{I}}_{\Pi}(\lambda)$:

```
1 $(x, y) \xleftarrow{\$} R$
  // The impersonator receives a polynomial number of honest transcripts and
     produces the first message of the protocol
2 $a[1] \xleftarrow{\$} \mathcal{I}^{\mathcal{O}^{\mathsf{TrGen}}}(x)$
  // The impersonator receives a random challenge for the round and produces the
     response until the end of the protocol
3 for $i \leftarrow 1$ to $\mu$ do
4     $\mathtt{ch}[i] \xleftarrow{\$} \mathsf{Ch}[i]$
5     $a[i+1] \xleftarrow{\$} \mathcal{I}(\{\mathtt{ch}[j]\}_{j\le i}, \{a[j]\}_{j\le i})$
6 end
  // The impersonator wins if the transcript produced verifies correctly
7 return $\mathsf{V}(y, a, \mathtt{ch})$
```

The oracle $\mathcal{O}^{\mathsf{TrGen}}$, to which the impersonator has access, produces valid transcripts for the statement x. We define $\mathcal{I}$'s advantage as their success probability, i.e.,

$$\mathbf{Adv}^{\mathcal{I}}_{\Pi}(\lambda) = \mathbb{P}(\mathsf{Exp}^{\mathcal{I}}_{\Pi}(\lambda) = 1).$$

We say that Π is *polynomially-secure against impersonations under passive attack* if $\mathbf{Adv}^{\mathcal{I}}_{\Pi}(\lambda)$ is negligible (in λ) for every probabilistic polynomial time impersonator $\mathcal{I}$.

It is possible to prove that any public-coin interactive proof which is knowledge-sound and (weak) Honest-Verifier Zero-Knowledge (wHVZK) is also polynomially-secure against impersonation under passive attack. In particular, we have the following:

Proposition 29. Let Π be a $(2\mu+1)$-round interactive proof system for a hard binary relation $R \subseteq X \times Y$ which is wHVZK and knowledge-sound with knowledge error κ. Let $\mathcal{I}$ be a passive impersonator against Π which makes Q queries to the transcript oracle $\mathcal{O}^{\mathsf{TrGen}}$. Then there exists an adversary $\mathcal{A}$ against the hard binary relation R such that

$$\mathbf{Adv}^{\Pi}_{\mathcal{I}} \le \mathtt{poly}(|x|) \cdot \mathbf{Adv}^{R}_{\mathcal{A}} + \kappa,$$

and the expected running time of **Adv** is about that of $\mathcal{I}$.

Theorem 30. Let Π be a $2\mu+1$-rounds interactive proof system which is secure against impersonation under passive attack. Then the signature scheme obtained by applying the Fiat-Shamir transform is EUF-CMA. In particular, the security loss introduced by the Fiat-Shamir transform is bounded by $\binom{Q}{\mu}$ where Q is the number of hash queries the adversary is allowed to do.

4.4 Security of the Signature Scheme

We proved in Proposition 23, 24 and 26 that CROSS-ID is complete, honest-verifier zero-knowledge and $(2,2)$-special sound. Thus, thanks to Theorem 27 we have that CROSS is knowledge-sound, with knowledge error:

$$\sum_{w'=\max\{0,w-t+\alpha\}}^{\min\{w,\alpha\}} \frac{\binom{\alpha}{w'}\binom{t-\alpha}{w-w'}}{\binom{t}{w}} \left(\frac{1}{p-1}\right)^{(\alpha-w')+(w-w')}.$$

By Proposition 29 we have that CROSS-ID is secure against impersonation under passive attack. Thus, thanks to Theorem 30, we have that CROSSis EUF-CMA secure, with a security loss of at most $\binom{Q}{2}$. These formulae, alongside the attack we show in the next section, are used to determine CROSS parameters.

Remark 31. In the definition of knowledge error, the extractor is required to be expected polynomial time and not *strict* polynomial time. Even if expected polynomial time is acceptable in many contexts [19, 17], it is often preferable to work in *strict* polynomial time since most of the hard problems are stated with respect to polynomial-time adversaries.

It turns out that both the extractors defined in [4, Lemma 3] and [1, Lemma 2], which are the basis of [5], work in expected polynomial time and are allowed to reach exponential time.

It is possible to show that both the extractors can be modified to be strict polynomial time, at the cost of a negligible loss in success probability, as shown in [5].

4.5 Forgery Attacks

In this section, we present two forgery attacks, derived from [5]. The former is adapted from [18] for weighted challenges, while the latter is a new attack. The choice of parameter sets is based on the complexity of the latter.

We conservatively estimate the cost of these forgeries in terms of CROSS operations. In our analysis, one elementary operation corresponds to simulating several of the instructions that the prover would perform. In particular, we conservatively identify the cost of a CROSS operation in 2^5 instructions (for details see 4.6). This allows us to easily assess the cost of such attacks so that the recommended CROSS instances meet the NIST security categories.

The first forgery we describe is relatively intuitive and attempts, for each round, to guess the first challenge $\texttt{chall}_1$ or the second challenge $\texttt{chall}_2$ (or both). The cost of this attack can be obtained as a direct corollary of the knowledge soundness of CROSS, but is derived for completeness in the following proposition.

Proposition 32. Since the protocol underlying CROSS is (t, w)-fixed-weight repetition of a $(2, 2)$-special sound interactive proof, a dishonest prover can convince a verifier if, for all executions, they either guess the first or the second individual challenge (or both) correctly. For the second challenges $\texttt{chall}_2[1], \dots, \texttt{chall}_2[t]$, if the adversary selects $\alpha \in \{0, \dots, t\}$ executions for the fixed-weight element, this attack runs in average time $O\left(\frac{1}{P_\alpha(t,w,p)}\right)$, where

$$P_\alpha(t, w, p) = \sum_{w'=\max\{0,w-t+\alpha\}}^{\min\{w,\alpha\}} \frac{\binom{\alpha}{w'}\binom{t-\alpha}{w-w'}}{\binom{t}{w}} \left(\frac{1}{p-1}\right)^{(\alpha-w')+(w-w')}.$$

The overall cost of the forgery is estimated by optimizing over $\alpha \in \{0, \dots, t\}$.

Proof. The forgery is successful if all rounds are accepted, that is, if for each round i either $\texttt{chall}_1[i]$ or $\texttt{chall}_2[i]$ (or both) have been guessed correctly. The average number of tests is given by the reciprocal of the probability that, for each round, both challenge values are correctly guessed.

Conservatively, we do not consider the cost of each test. Still, we lower bound the cost of the forgery by using the average number of tests before the adversary's guesses are valid for each round.

Let us consider t rounds and a fixed-weight second challenge $\texttt{chall}_2 = \big(\texttt{chall}_2[i], \dots, \texttt{chall}_2[t]\big)$, with $\texttt{chall}_2[i] \in \{0, 1\}$ being the second challenge for the i-th round. When $\texttt{chall}_2$ has weight w, there will exist w many rounds with $\texttt{chall}_2[i] = 1$ and $t - w$ many rounds with $\texttt{chall}_2[i] = 0$.

The adversary could now guess the w rounds of $\texttt{chall}_2[i] = 1$. Notice, however, that a better strategy when $w \neq t/2$ is to select $\alpha \in \{0, \dots, t\}$ rounds of $\texttt{chall}_2[i] = 1$.

If the adversary chooses a challenge in one round correctly, this round will be accepted. However, if the adversary picks a challenge wrong, there is still the possibility of having chosen $\texttt{chall}_1[i]$ correctly. Thus, let us assume that w' many rounds out of the α guessed $\texttt{chall}_2[i] = 1$-rounds are correct, for $w' \in \{\max\{0, w - t + \alpha\}, \dots, \min\{w, \alpha\}\}$.

This implies that there are $\alpha - w'$ mistakes in the guessed $\texttt{chall}_2[i] = 0$-rounds and $w - w'$ mistakes in the guessed $\texttt{chall}_2[i] = 1$-rounds. For each error, the adversary must have guessed the corresponding $\texttt{chall}_1[i]$ correctly.

For a fixed α, this gives an overall cheating probability of

$$\sum_{w'=\max\{0,w-t+\alpha\}}^{\min\{w,\alpha\}} \frac{\binom{\alpha}{w'}\binom{t-\alpha}{w-w'}}{\binom{t}{w}} \left(\frac{1}{p-1}\right)^{(\alpha-w')+(w-w')}.$$

□

We now report a new attack described in [5]. The attack is based on the forgery described in [18] and later optimized for CROSS for fixed-weight challenges. The attack makes use of the fact that the second challenge is generated after the first challenge, and, furthermore, it is possible to generate multiple second challenges without modifying the commitments or the first challenge value.

This way, one can split the forgery into two separate phases, where the overall cost is given by the sum of the two associated costs. As in the interactive case, the fixed-weight distribution of the second challenge can be further exploited to optimize the round selection.

The attack is described in [5] in full generality for $q2$-identification schemes. Below we provide the detailed application of the algorithm to CROSS with an updated complexity estimate that takes into account the cost of unitary operations.

For simplicity, we leave several steps of the actual protocol away, e.g., an impersonator would not choose $\mathbf{v}[i] \in G$ but rather $\overline{\mathbf{v}}_G[i]$ and then compute $\mathbf{v}[i]$.

Proposition 33. We consider the following procedure:

1) sample $\texttt{Salt} \xleftarrow{\$} \{0,1\}^{2\lambda}$, $\texttt{Seed} \xleftarrow{\$} \{0,1\}^{\lambda}$, generate seeds $\texttt{Seed}[1], \dots, \texttt{Seed}[t]$ using the PRNG tree;
2) guess values $\texttt{chall}'_1 = \big(\texttt{chall}'_1[1], \dots, \texttt{chall}'_1[t]\big)$ for the first challenge;
3) guess values $\texttt{chall}'_2 = \big(\texttt{chall}'_2[1], \dots, \texttt{chall}'_2[t]\big)$ for the second challenge, choosing $\alpha \geq w$ rounds for $\texttt{chall}_2[i] = 1$ (consider that $\texttt{chall}'_2$ has weight w);
4) for each $i = 1, \dots, t$, do:
    - 4.1) sample $\mathbf{u}'^{(i)} \in \mathbb{F}_p^n$ and $\mathbf{e}'^{(i)} \in G$ using $\texttt{Seed}[i]$;
    - 4.2) choose an arbitrary $\mathbf{v}[i] \in G$;
    - 4.3) compute $\mathbf{y}^*[i] = \mathbf{u}'[i] + \texttt{chall}'_1[i]\mathbf{e}'[i]$;
    - 4.4) compute $\mathbf{s}'[i] = (\mathbf{v}[i] \star \mathbf{y}^*[i])\mathbf{H}^\top$;
    - 4.5) set $\texttt{cmt}_0[i] = \mathsf{Hash}\big(\mathbf{s}'[i] - \texttt{chall}'_1[i]\mathbf{s}, \mathbf{v}[i], \texttt{Salt}, i\big)$;
    - 4.6) set $\texttt{cmt}_1[i] = \mathsf{Hash}(\mathbf{u}'[i], \mathbf{e}'[i], \texttt{Salt}, i)$;
5) compute $\texttt{digest}_{\texttt{cmt}_0}$ as the root of the tree with leaves $\texttt{cmt}_0[i], \dots, \texttt{cmt}_0[t]$ and $\texttt{digest}_{\texttt{cmt}_1} = \mathsf{Hash}\big(\texttt{cmt}_1[1] \mid \cdots \mid \texttt{cmt}_1[t]\big)$;

   compute $\texttt{digest}_{\texttt{cmt}} = \mathsf{Hash}(\texttt{digest}_{\texttt{cmt}_0} \mid \texttt{digest}_{\texttt{cmt}_1})$ and $\texttt{digest}_{\texttt{Msg}} = \mathsf{Hash}(\texttt{Msg})$;

   compute $\texttt{digest}_{\texttt{chall}_1} = \mathsf{Hash}(\texttt{digest}_{\texttt{Msg}} \mid \texttt{digst}_{\texttt{cmt}} \mid \texttt{Salt})$;

   generate $\big(\texttt{chall}_1[1], \dots, \texttt{chall}_1[t]\big) = \mathsf{CSPRNG}(\texttt{digest}_{\texttt{chall}_1} \mid t + c)$;
6) let $S = \{i \in \{1, \dots, t\} \,|\texttt{chall}'_1[i] = \texttt{chall}_1[i]\}$; if $|S| \geq t^*$, proceed. Otherwise, restart from step 1);
7) for each round $i \in S$ (i.e., such that $\texttt{chall}_1[i] = \texttt{chall}'_1[i]$), set $\mathbf{y}[i] = \mathbf{y}^*[i]$;
8) for each round $i \notin S$ (i.e., such that $\texttt{chall}_1[i] \neq \texttt{chall}'_1[i]$), do:
    - 8.1) if $\texttt{chall}'_2[i] = 0$:
        - S0.1) choose $\widetilde{\mathbf{e}}[i] \in \mathbb{F}_p^n$ such that $\widetilde{\mathbf{e}}[i]\mathbf{H}^\top = \mathbf{s}$;
        - S0.2) choose $\widetilde{\mathbf{u}}[i] \in \mathbb{F}_p^n$ such that $\widetilde{\mathbf{u}}[i]\mathbf{H}^\top = (\mathbf{v}[i] \star \mathbf{y}^*[i])\mathbf{H}^\top - \texttt{chall}_1[i]\mathbf{s}$;
        - S0.3) set $\mathbf{y}[i] = \widetilde{\mathbf{u}}[i] + \texttt{chall}_1[i]\widetilde{\mathbf{e}}[i]$;
    - 8.2) if $\texttt{chall}'_2[i] = 1$:
        - S1.1) set $\mathbf{y}[i] = \mathbf{y}^*[i]$;
9) compute $\texttt{digest}_{\texttt{chall}_2} = \mathsf{Hash}(\mathbf{y}[1] \mid \cdots \mid \mathbf{y}[t] \mid \texttt{digest}_{\texttt{chall}_1})$ and generate $\big(\texttt{chall}_2[1], \dots, \texttt{chall}_2[t]\big) = \mathsf{Hash}(\texttt{digest}_{\texttt{chall}_2} \mid t + c + 1)$;

Table 4: Bit-complexity estimates for signature forgery with parameters as used by CROSS. Values t^* and α shows the optimal choice of attack parameters.

Algorithm and Security Category	**Optim. Corner**	**Parameters**			**Forgery Params.**		**Cost (bit)**
		p	t	w	t^*	α	
CROSS-R-SDP **1**	fast	127	157	82	33	83	128.11
	balanced	127	256	215	39	231	128.04
	small	127	520	488	50	512	128.06
CROSS-R-SDP **3**	fast	127	239	125	51	127	192.01
	balanced	127	384	321	60	345	192.43
	small	127	580	527	68	559	192.11
CROSS-R-SDP **5**	fast	127	321	167	69	169	256.02
	balanced	127	512	427	80	459	256.35
	small	127	832	762	95	807	256.31
CROSS-R-SDP(G) **1**	fast	509	147	76	23	76	128.05
	balanced	509	256	220	26	231	128.07
	small	509	512	484	32	499	128.93
CROSS-R-SDP(G) **3**	fast	509	224	119	36	120	192.05
	balanced	509	268	196	37	206	192.11
	small	509	512	463	44	483	193.28
CROSS-R-SDP(G) **5**	fast	509	300	153	48	154	256.06
	balanced	509	356	258	50	271	256.34
	small	509	642	575	58	600	256.02

10) if $\big(\texttt{chall}_2[1], \dots, \texttt{chall}_2[t]\big)$ and $\big(\texttt{chall}'_2[1], \dots, \texttt{chall}'_2[t]\big)$ are equal for all indices $i \notin S$, proceed. Otherwise, restart from step 8);

11) compute $\texttt{Proof}$ which allows to recover $\texttt{digest}_{\texttt{cmt}_0}$ and $\texttt{Path}$ which allows to recover $\texttt{Seed}[i]$ for all i with $\texttt{chall}_2[i] = 1$.

12) for each $i = 1, \dots, t$: if $\texttt{chall}_2[i] = 0$, set $\texttt{resp}[i]_0 = (\mathbf{y}[i], \mathbf{v}[i])$ and $\texttt{resp}[i]_1 = \texttt{cmt}_1[i]$.

The above algorithm is a forgery running on average time

$$O\left(\min_{t^*\in\{0,\dots,t\}}\left\{\frac{1}{P_1(t,t^*,p)} + \frac{1}{P_2(t,t^*,w,p)}\right\}\right),$$

where

$$P_1(t,t^*,p) = \sum_{j=t^*}^{t}\binom{t}{j}\left(\frac{1}{p-1}\right)^{j}\left(1-\frac{1}{p-1}\right)^{t-j},$$

$$P_2(t,t^*,w,p) = \max_{\alpha\in\{w,\dots,t\}}\sum_{j=t^*}^{t}\frac{\binom{t}{j}\left(\frac{1}{p-1}\right)^{j}\left(1-\frac{1}{p-1}\right)^{t-j}}{P_1(t,t^*,p)}\sum_{w^*=\max\{0,\alpha-j\}}^{\min\{t-j,w\}}\frac{\binom{t-j}{w^*}\binom{j}{\alpha-w^*}}{\binom{t}{\alpha}}\frac{\binom{j}{w-w^*}}{\binom{t}{w}}.$$

Proof. The strategies followed by the forgery are, essentially, a rewriting of the ones in the proof for Proposition 26, with the additional advantage of having the second challenge with fixed-weight w.

The algorithm iterates over the first loop (steps 1–6) until the choices on the first challenge are valid for at least t^* rounds.

Once this is obtained, the algorithm freezes the commitments, and the first challenge then starts making attempts until the second challenge is correctly generated. This is the purpose of steps 7–10.

For each attempt, the algorithm uses fresh values $\widetilde{\mathbf{e}}[i]$: this leads to a different $\mathbf{y}[i]$ and, consequently, to new values for the second challenge $\big(\texttt{chall}_2[1], \dots, \texttt{chall}_2[t]\big)$. By doing this, the commitments prepared in the initial loop remain valid.

This procedure gets repeated until the second challenge amends the situation; namely, in every round where the attacker did not guess the correct value for the first challenge, the value for the second challenge must be correct.

The total cost of the attack is the sum of the costs for the two phases. The probability that the guess $\left(\mathtt{chall}'_1[1], \ldots, \mathtt{chall}'_1[t]\right)$ i matches in at least t^* positions with $\left(\mathtt{chall}_1[1], \ldots, \mathtt{chall}_1[t]\right)$, is

$$P_1(t, t^*, p) = \sum_{j=t^*}^{t} \binom{t}{j} \left(\frac{1}{p-1}\right)^j \left(1 - \frac{1}{p-1}\right)^{t-j}.$$

Consequently, the average cost for the first loop is $O\left(\frac{1}{P_1(t,t^*,p)}\right)$.

We now consider the second loop. Let S denote the set of indices i for which $\mathtt{chall}_1[i] = \mathtt{chall}'_1[i]$ and its complement by S^C. In the following, we will indicate $j = |S|$; notice that[1]

$$\mathbb{P}[|S| = j] = \frac{\binom{t}{j}\left(\frac{1}{p-1}\right)^j \left(1 - \frac{1}{p-1}\right)^{t-j}}{P_1(t, t^*, p)}.$$

Let $\mathtt{chall}_2 = \left(\mathtt{chall}_2[1], \ldots, \mathtt{chall}_2[t]\right)$ and $\mathtt{chall}'_2 = \left(\mathtt{chall}'_2[1], \ldots, \mathtt{chall}'_2[t]\right)$, and indicate by $\mathtt{chall}_2[S]$ (resp., $\mathtt{chall}'_2[S]$) the vector formed by the coordinates of $\mathtt{chall}_2$ (resp., $\mathtt{chall}'_2$) which are indexed by S. Analogously, we denote by $\mathtt{chall}_2[S^C]$ (resp., $\mathtt{chall}'_2[S^C]$) the vector formed by the coordinates of $\mathtt{chall}_2$ (resp., $\mathtt{chall}'_2$) which are not indexed by S.

For the second loop to halt, $\mathtt{chall}_2$ must be such that $\mathtt{chall}_2[S^C] = \mathtt{chall}'_2[S^C]$. Let w^* denote the number of 1-guesses for the rounds indexed by S^C; that is, w^* is the Hamming weight of $\mathtt{chall}'_2[S^C]$.

Recall that, in guessing $\mathtt{chall}'_2$ the adversary chooses $\alpha \geq w$ position for the 1-entries. It follows that

$$\mathbb{P}\left[\mathrm{wt}(\mathtt{chall}'_2[S^C]) = w^*\right] = \frac{\binom{t-j}{w^*}\binom{j}{\alpha-w^*}}{\binom{t}{\alpha}}.$$

The probability that a generated $\mathtt{chall}_2$ is valid, i.e., $\mathtt{chall}_2[S^C] = \mathtt{chall}'_2[S^C]$, is

$$\begin{aligned}
&\mathbb{P}\left[\mathtt{chall}_2 \text{ is valid} \,\middle|\, \mathrm{wt}(\mathtt{chall}'_2[S^C]) = w^*\right] \\
&= \frac{\left|\left\{\mathtt{chall}_2 \in \{0,1\}^t \,\middle|\, \mathrm{wt}(\mathtt{chall}'_2[S^C]) = w^*,\ \mathtt{chall}_2[S^C] = \mathtt{chall}'_2[S^C]\right\}\right|}{\binom{t}{w}} \\
&= \frac{\left|\left\{\mathtt{chall} \in \{0,1\}^t \,\middle|\, \mathrm{wt}(\mathtt{chall}_2[S]) = w - w^*,\ \mathtt{chall}_2[S^C] = \mathtt{chall}'_2[S^C]\right\}\right|}{\binom{t}{w}} \\
&= \frac{\binom{j}{w-w^*}}{\binom{t}{w}}.
\end{aligned}$$

An example of a valid $\mathtt{chall}_2$ is reported below, for example, with $t^* = 5$, $\alpha = t - 1$ and $w^* = t - t^* - 1$.

	First t^* rounds					Last $t - t^*$ rounds							
Guessed $\mathtt{chall}'_1[i]$	2	71	16	23	4	5	98	121	46	29	82	$\cdots$	45
Guessed $\mathtt{chall}'_2[i]$	1	1	1	1	1	1	0	1	1	1	1	$\cdots$	1
Actual $\mathtt{chall}_1[i]$	2	71	16	23	4	7	120	99	21	7	124	$\cdots$	3
Actual $\mathtt{chall}_2[i]$	1	1	0	1	1	1	0	1	1	1	1	$\cdots$	1
	Weight $w - w^*$					Weight w^*							

[1]Formally this is the conditional probability, given that $|S| \geq t^*$; to avoid burdening the (already involved) notation, we do not indicate it explicitly.

Putting everything together, we have that a test for $\texttt{chall}_2$ is valid with an average probability

$$\begin{aligned}P_2(t,t^*,w,p) &= \max_{\alpha\in\{w,\dots,t\}} \sum_{j=t^*}^{t} \mathbb{P}[|S|=j] \\ &\cdot \sum_{w^*=\max\{0,\alpha-j\}}^{\min\{t-j,\alpha\}} \mathbb{P}[\mathrm{wt}(\texttt{chall}'_2[S^C])=w^*]\cdot \mathbb{P}\left[\texttt{chall}_2 \text{ is valid } \middle| \mathrm{wt}(\texttt{chall}'_2[S^C]=w^*)\right] \\ &= \max_{\alpha\in\{w,\dots,t\}} \sum_{j=t^*}^{t} \frac{\binom{t}{j}\left(\frac{1}{p-1}\right)^j\left(1-\frac{1}{p-1}\right)^{t-j}}{P_1(t,t^*,p)} \sum_{w^*=\max\{0,\alpha-j\}}^{\min\{t-j,w\}} \frac{\binom{t-j}{w^*}\binom{j}{\alpha-w^*}}{\binom{t}{\alpha}}\frac{\binom{j}{w-w^*}}{\binom{t}{w}}.\end{aligned}$$

The overall cost of the attack is estimated by summing the costs for both phases and optimizing over t^*, that is

$$\min_{t^*\in\{0,\dots,t\}}\left\{\frac{1}{P_1(t,t^*,p)}+\frac{1}{P_2(t,t^*,w,p)}\right\}. \qquad \square$$

Expected security strength: Table 4 presents the computational cost of the forgery attack for parameter sets as used by CROSS.

We now show that we can target a smaller forgery cost (by 5 bits), as each forgery attempt makes several calls to SHAKE.

4.6 Finite Regime Cost of a Forgery

Considering the aforementioned cost of forging a signature, we have that the big-O notation hides the constant cost of actually performing the computations required to perform a forgery attempt.

A straightforward, but effective, lower bound to the said constant cost is represented by a single call to the Hash· function, which we denote as c_{Hash}.

The security requirements for NIST category 1, 3 and 5 are stated with respect to the hardness of breaking AES with a 128, 192, and 256 bit long key, respectively. This corresponds to $2^\ell c_{\mathsf{AES}-\ell}$, where ℓ is the length in bits of the AES key, and $c_{\mathsf{AES}-\ell}$ is the cost of a single computation of AES with an ℓ bit key, expressed as the count of Boolean gates.

As a consequence, we have that the maximum tolerable forgery probability x can be conservatively estimated by solving the equation $2^\ell c_{\mathsf{AES}-\ell} = \frac{1}{x}c_{\mathsf{Hash}}$ for x. This in turn yields $x = 2^{-\ell}\frac{c_{\mathsf{Hash}}}{c_{\mathsf{AES}-\ell}}$.

Therefore, if the cost of a hash call, quantified in Boolean gates, is higher than the one of an AES execution with the appropriate key length, the tolerable forgery probability will be higher than $2^{-\ell}$ by a factor $\frac{c_{\mathsf{Hash}}}{c_{\mathsf{AES}-\ell}}$.

In our case, Hash· is implemented with the NIST standard SHAKE [23] algorithms, which has its complexity dominated by the computation of the inner Keccak-f function, repeated 24 times for each ingested block of input. Providing a quantitative lower bound for the Boolean gate cost of SHAKE, can be done assuming the shortest possible input, and counting the number of Boolean gates to realize f.

We note that minimizing the Boolean complexity of f is very simple (due to its small nature), and a gate count yields $\approx$ 736kgates. We note that this gate count does not consider the cost of writing the SHAKE state in the chosen memory technology, nor the delays of the wiring, providing a definitely conservative estimate of the SHAKE cost.

Considering the estimated AES cost in Boolean gates reported in the NIST call [24], can be derived to be 2^{15} for AES-128 (as the document reports $2^\ell c_{\mathsf{AES}-\ell} = 2^{143}$) we have that $\frac{c_{\mathsf{Hash}}}{c_{\mathsf{AES}-\ell}} \approx 22.4$, in turn allowing us to raise the maximum forgery probability to $2^{-\ell+\log_2(22.4)} \approx 2^{-\ell+4.489}$.

We round the figure to $2^{-\ell+5}$, as the remaining cost of the computations required to perform a forgery attempt are at least as demanding as a SHAKE computation.

5 Bibliography

[1] Thomas Attema and Serge Fehr. Parallel repetition of $(k_1, \ldots, k_\mu)$-special-sound multi-round interactive proofs. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, pages 415–443, Cham, 2022. Springer Nature Switzerland.

[2] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. A finite regime analysis of information set decoding algorithms. *Algorithms*, 12(10):209, 2019.

[3] Marco Baldi, Sebastian Bitzer, Alessio Pavoni, Paolo Santini, Antonia Wachter-Zeh, and Violetta Weger. Zero knowledge protocols and signatures from the restricted syndrome decoding problem. *PKC 2024*, 2024.

[4] Michele Battagliola, Riccardo Longo, Federico Pintore, Edoardo Signorini, and Giovanni Tognolini. Security of fixed-weight repetitions of special-sound multi-round proofs. Cryptology ePrint Archive, Paper 2024/884, 2024.

[5] Michele Battagliola, Riccardo Longo, Federico Pintore, Edoardo Signorini, and Giovanni Tognolini. A revision of CROSS security: Proofs and attacks for multi-round fiat-shamir signatures. Cryptology ePrint Archive, Paper 2025/127, 2025.

[6] Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved generic algorithms for hard knapsacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 364–385. Springer, 2011.

[7] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How 1+ 1= 0 improves information set decoding. In *Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31*, pages 520–536. Springer, 2012.

[8] Daniel J Bernstein, Tanja Lange, and Christiane Peters. Smaller decoding exponents: ball-collision decoding. In *Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings 31*, pages 743–760. Springer, 2011.

[9] Ward Beullens, Pierre Briaud, and Morten Øygarden. A security analysis of restricted syndrome decoding problems. Cryptology ePrint Archive, Paper 2024/611, 2024.

[10] Sebastian Bitzer, Alessio Pavoni, Violetta Weger, Paolo Santini, Marco Baldi, and Antonia Wachter-Zeh. Generic decoding of restricted errors. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 246–251. IEEE, 2023.

[11] Il'ya Isaakovich Dumer. Two decoding algorithms for linear codes. *Problemy Peredachi Informatsii*, 25(1):24–32, 1989.

[12] Andre Esser, Alexander May, and Floyd Zweydinger. McEliece needs a break–solving McEliece-1284 and quasi-cyclic-2918 with modern isd. In *Advances in Cryptology–EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30–June 3, 2022, Proceedings, Part III*, pages 433–457. Springer, 2022.

[13] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In *Advances in Cryptology–ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings 15*, pages 88–105. Springer, 2009.

[14] Cheikh Thiécoumba Gueye, Jean Belo Klamti, and Shoichi Hirose. Generalization of BJMM-ISD using May-Ozerov nearest neighbor algorithm over an arbitrary finite field $\mathbb{F}_q$. In *International Conference on Codes, Cryptology, and Information Security*, pages 96–109. Springer, 2017.

[15] Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *Journal of the ACM (JACM)*, 21(2):277–292, 1974.

[16] Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In *Advances in Cryptology – EUROCRYPT 2010*, pages 235–256. Springer, 2010.

[17] Joseph Jaeger and Stefano Tessaro. Expected-time cryptography: Generic techniques and applications to concrete soundness. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography*, pages 414–443, Cham, 2020. Springer International Publishing.

[18] Daniel Kales and Greg Zaverucha. An attack on some signature schemes constructed from five-pass identification schemes. In *Cryptology and Network Security: 19th International Conference, CANS 2020, Vienna, Austria, December 14–16, 2020, Proceedings*, pages 3–22. Springer, 2020.

[19] Jonathan Katz and Yehuda Lindell. Handling expected polynomial-time strategies in simulation-based security proofs. *SSRN Electronic Journal*, 01 2006.

[20] Shai Levin. A note on zero-knowledge simulator of the CROSS identification protocol. Cryptology ePrint Archive, Paper 2025/359, 2025.

[21] Felice Manganiello and Freeman Slaughter. Generic error SDP and generic error CVE. In Andre Esser and Paolo Santini, editors, *Code-Based Cryptography*, pages 125–143, Cham, 2023. Springer Nature Switzerland.

[22] Alexander Meurer. *A coding-theoretic approach to cryptanalysis.* PhD thesis, Ruhr-Universität Bochum, 2013.

[23] National Institute of Standards and Technology. FIPS 202 - SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf, 2015.

[24] National Institute of Standards and Technology. Post-quantum crypto project, December 2016.

[25] Paolo Santini, Marco Baldi, and Franco Chiaraluce. Computational hardness of the permuted kernel and subcode equivalence problems. *IEEE Transactions on Information Theory*, 2023.

[26] Jacques Stern. A method for finding codewords of small weight. In *International Colloquium on Coding Theory and Applications*, pages 106–113. Springer, 1988.

[27] David Wagner. A generalized birthday problem. In *Annual International Cryptology Conference*, pages 288–304. Springer, 2002.

[28] Violetta Weger, Karan Khathuria, Anna-Lena Horlemann, Massimo Battaglioni, Paolo Santini, and Edoardo Persichetti. On the hardness of the Lee syndrome decoding problem. *Advances in Mathematics of Communications*, 2022.

A Proof of NP-Completeness

This section provides a proof of Theorem 2, i.e., it is shown that R-SDP is NP-complete. Note that this result can also be inferred from [28], where the authors show that the additivity of the weight and having a unit in the ambient space of the error vector is enough to obtain NP-completeness. For the sake of completeness, we state the proof in the following.

Proof. Recall the NP-hard 3-Dimensional Matching (3DM) problem, where one is given the instance $T = \{b_1, \dots, b_t\}$, with $|T| = t, U \subset T \times T \times T$ and $|U| = u$ and asks whether there exists a $W \subset U$ with $|W| = t$ and no two words in W coincide in any position.

Recall that the original SDP has a reduction from 3DM through the following construction: let $\mathbf{H} \in \mathbb{F}_p^{(3t)\times u}$ be the incidence matrix, i.e., each column of $\mathbf{H}$ corresponds to a word in U and the rows correspond to $T \times T \times T$; thus, the rows $\{1, \dots, t\}$ correspond to the first position of the word $\mathbf{u}$, the rows $\{t+1, \dots, 2t\}$ correspond to the second position of $\mathbf{u}$ and the rows $\{2t+1, \dots, 3t\}$ correspond to the third position of $\mathbf{u}$. More formally, let $T = \{b_1, \dots, b_t\}$, $U = \{\mathbf{a}_1, \dots, \mathbf{a}_u\}$ and

- for $i \in \{1, \dots, t\}$, we set $h_{i,j} = 1$ if $\mathbf{a}_j[1] = b_i$ and $h_{i,j} = 0$ else,

- for $i \in \{t+1, \dots, 2t\}$, we set $h_{i,j} = 1$ if $\mathbf{a}_j[2] = b_i$ and $h_{i,j} = 0$ else,
- for $i \in \{2t+1, \dots, 3t\}$, we set $h_{i,j} = 1$ if $\mathbf{a}_j[3] = b_i$ and $h_{i,j} = 0$ else.

We also set $\mathbf{s} \in \mathbb{F}_p^{3t}$ to be the all-one vector.

From the original reduction, we know that any solution $\mathbf{e} \in \mathbb{F}_p^u$ with $\mathbf{H}\mathbf{e}^\top = \mathbf{s}^\top$ has weight t (it has weight at least t as we need to reach the all-one vector in $\mathbb{F}_p^{3t}$ and each column gives weight 3, and it has weight at most t as p is larger than u and else we would get syndrome entries larger than 1) and its support corresponds to the solution W. That is, the columns of $\mathbf{H}$ indexed by the support of $\mathbf{e}$ are the t words in W.

The polynomial reduction from 3DM to R-SDP uses this construction as well. Let T of size t and $U \subset T \times T \times T$ of size u be an instance of 3DM. Let $\mathbf{H} \in \mathbb{F}_p^{(3t)\times u}$ be the incidence matrix and let

$$\widetilde{\mathbf{H}} = \begin{pmatrix} \mathbf{H} & -g \star \mathbf{H} \\ \mathrm{Id}_u & \mathrm{Id}_u \end{pmatrix} \in \mathbb{F}_p^{(3t+u)\times 2u}$$

be a parity-check matrix. Let us consider the syndrome $(\mathbf{s}, \mathbf{s}') \in \mathbb{F}_p^{3t+u}$ with $\mathbf{s} = (1-g^2, \dots, 1-g^2) \in \mathbb{F}_p^{3t}$ and $\mathbf{s}' = (1+g, \dots, 1+g) \in \mathbb{F}_p^u$. Thus, the instance of R-SDP given by $\widetilde{\mathbf{H}}$ and $(\mathbf{s}, \mathbf{s}')$ is asking for $(\mathbf{e}, \mathbf{e}') \in \mathbb{E}^{2u}$ such that

$$(\mathbf{e}, \mathbf{e}')\widetilde{\mathbf{H}}^\top = (\mathbf{s}, \mathbf{s}'),$$

where $\mathbb{E} = \{g^i \mid i \in \{0, \dots, z-1\}\}$. By assumption of R-SDP, we use a g of order $2 < z < p-1$.

We consider two cases.

1. Assume that the R-SDP solver returns "yes", i.e., there exists $\mathbf{e}, \mathbf{e}' \in \mathbb{E}^u$ such that $(\mathbf{e}, \mathbf{e}')\widetilde{\mathbf{H}}^\top = (\mathbf{s}, \mathbf{s}')$. Hence,

$$\mathbf{H}\mathbf{e}^\top - g \star \mathbf{H}\mathbf{e}'^\top = (1-g^2, \dots, 1-g^2)^\top,$$
$$\mathbf{e} + \mathbf{e}' = (1+g, \dots, 1+g).$$

   Hence, for each $i \in \{1, \dots, u\}$ we have $e_i + e'_i = 1 + g$.

   Let us assume (we later show that this hypothesis is not needed, but it facilitates the proof) that the only elements in $\mathbb{E}$ that add to $1+g$ are 1 and g.

   Hence, whenever $e_i = 1$, we must have $e'_i = g$ and whenever $e_i = g$, we must have $e'_i = 1$. Thus, we split $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_g$ and $\mathbf{e}' = \mathbf{e}'_1 + \mathbf{e}'_g$ where $\mathbf{e}_1, \mathbf{e}'_1 \in \{0,1\}^u, \mathbf{e}_g, \mathbf{e}'_g \in \{0, g\}^u$ and $\mathrm{supp}(\mathbf{e}_1) = S = \mathrm{supp}(\mathbf{e}'_g)$ and $\mathrm{supp}(\mathbf{e}'_1) = S^C = \mathrm{supp}(\mathbf{e}_g)$. This implies that $\mathbf{e}_g = g \star \mathbf{e}'_1$ and $\mathbf{e}'_g = g \star \mathbf{e}_1$.

   The first parity-check equation can now be reformulated as

$$\begin{aligned} &\mathbf{H}\mathbf{e}^\top - g \star \mathbf{H}\mathbf{e}'^\top \\ =&\mathbf{H}\mathbf{e}_1^\top - g \star \mathbf{H}\mathbf{e}_g'^\top + \mathbf{H}\mathbf{e}_g^\top - g \star \mathbf{H}\mathbf{e}_1'^\top \\ =&\mathbf{H}\mathbf{e}_1^\top - g^2 \star \mathbf{H}\mathbf{e}_1^\top + g \star \mathbf{H}\mathbf{e}_1'^\top - g \star \mathbf{H}\mathbf{e}_1'^\top \\ =&(1-g^2) \star \mathbf{H}\mathbf{e}_1^\top \\ =&(1-g^2, \dots, 1-g^2) = \mathbf{s}' \ ; \end{aligned}$$

   thus, $\mathbf{H}\mathbf{e}_1^\top = (1, \dots, 1)$ is such that $\mathrm{supp}(\mathbf{e}_1)$ corresponds to a solution W of 3DM, as in the classical reduction.

2. Assume that the R-SDP solver returns "no", i.e., there exists no $\mathbf{e}, \mathbf{e}' \in \mathbb{E}^u$ such that $(\mathbf{e}, \mathbf{e}')\widetilde{\mathbf{H}}^\top = (\mathbf{s}, \mathbf{s}')$.

   Let us assume by contradiction, that the 3DM has a solution W. We can then define S to be the indices of words in U belonging to the solution W. Let us define $\mathbf{e}_1, \mathbf{e}'_1 \in \{0,1\}^u, \mathbf{e}_g, \mathbf{e}'_g \in \{0, g\}^u$ with $\mathrm{supp}(\mathbf{e}_1) = S = \mathrm{supp}(\mathbf{e}'_g)$ and $\mathrm{supp}(\mathbf{e}'_1) = S^C = \mathrm{supp}(\mathbf{e}_g)$.

   From this, it also follows that $\mathbf{e}_g = g \star \mathbf{e}'_1$ and $\mathbf{e}'_g = g \star \mathbf{e}_1$. Then, the vector $(\mathbf{e}_1 + \mathbf{e}_g, \mathbf{e}'_1 + \mathbf{e}'_g) \in \mathbb{E}^{2u}$ is a solution to the R-SDP, as in case 1, which gives the desired contradiction, to the R-SDP solver returning "no".

Note that the hypothesis that only 1 and g in $\mathbb{E}$ add up to $1+g$ is not necessary.

For this assume that there exists $g^i, g^j \in \mathbb{E}$, with $0 \neq i < j < z$ such that $g^i + g^j = 1 + g$. Then, the splitting of $\mathbf{e}$ and $\mathbf{e}'$ is a bit more complicated:

$$\begin{aligned}\mathbf{e} &= \mathbf{e}_1 + \mathbf{e}_g + \mathbf{e}_i + \mathbf{e}_j,\\ \mathbf{e}' &= \mathbf{e}'_1 + \mathbf{e}'_g + \mathbf{e}'_i + \mathbf{e}'_j,\end{aligned}$$

where $\mathbf{e}_1, \mathbf{e}'_1 \in \{0,1\}^u$,$\mathbf{e}_g, \mathbf{e}'_g \in \{0,g\}^u$,$\mathbf{e}_i, \mathbf{e}'_i \in \{0,g^i\}^u$,$\mathbf{e}_j, \mathbf{e}'_j \in \{0,g^j\}^u$ with

$$\begin{aligned}\mathrm{supp}(\mathbf{e}_1) &= S_1 = \mathrm{supp}(\mathbf{e}'_g),\\ \mathrm{supp}(\mathbf{e}_g) &= S'_1 = \mathrm{supp}(\mathbf{e}'_1),\\ \mathrm{supp}(\mathbf{e}_i) &= S_i = \mathrm{supp}(\mathbf{e}'_j),\\ \mathrm{supp}(\mathbf{e}_j) &= S'_i = \mathrm{supp}(\mathbf{e}'_i),\end{aligned}$$

and the supports S_1, S'_1, S_i, S'_i are distinct and partition $\{1, \ldots, u\}$. Again, it follows that

$$\begin{aligned}\mathbf{e}_g &= g \star \mathbf{e}'_1,\\ \mathbf{e}'_g &= g \star \mathbf{e}_1,\\ \mathbf{e}_j &= g^{j-i} \star \mathbf{e}'_i,\\ \mathbf{e}'_j &= g^{j-i} \star \mathbf{e}_i.\end{aligned}$$

Thus, by rewriting the first parity-check equation, we get

$$\begin{aligned}&\mathbf{He}^\top - g \star \mathbf{He}'^\top\\ =&\mathbf{He}_1^\top + \mathbf{He}_g^\top + \mathbf{He}_i^\top + \mathbf{He}_j^\top\\ &- g \star \mathbf{He}_1'^\top - g \star \mathbf{He}_g'^\top - g \star \mathbf{He}_i'^\top - g \star \mathbf{He}_j'^\top\\ =&\mathbf{He}_1^\top + g \star \mathbf{He}_1'^\top + \mathbf{He}_i^\top + g^{j-i} \star \mathbf{He}_i'^\top\\ &- g \star \mathbf{He}_1'^\top - g^2 \star \mathbf{He}_1^\top - g \star \mathbf{He}_i'^\top - g^{j-i+1} \star \mathbf{He}_i^\top\\ =&(1-g^2) \star \mathbf{He}_1^\top + (1-g^{j-i+1}) \star \mathbf{He}_i^\top + (g^{j-i} - g) \star \mathbf{He}_i'^\top\\ =&(1-g^2, \ldots, 1-g^2) = \mathbf{s}'.\end{aligned}$$

Since $\mathbf{e}_1, \mathbf{e}_i, \mathbf{e}'_i$ all have different supports, the only way to get $1-g^2$ in each entry is to have $\mathbf{e}_i = \mathbf{e}'_i = 0$. In fact, any other sum leads to a contradiction:

- If $(1-g^2) + (1-g^{j-i+1}) = 1-g^2$ then $1 = g^{j-i+1}$ and hence $j = i-1$ which contradicts $j > i$.
- If $(1-g^2) + (g^{j-i} - g) = 1-g^2$ then $g^{j-i} = g$ and hence $j - i = 1$. However, as then $g^j + g^i = g^i(1+g) = 1+g$, it follows that $g^i = 1$, which contradicts $i \neq 0$.
- If $(1-g^2) + (1-g^{j-i+}) + (g^{j-i} - g) = 1-g^2$, then $1 + g^{j-i} = g^{j-i+1} + g = g(1+g^{j-i})$ and thus $g = 1$, which contradicts $\mathbb{E} \neq \mathbb{F}_q^\star$.
- If $(1-g^{j-i+1}) + (g^{j-i} - g) = 1-g^2$, then $g^{j-i} - g^{j-i+1} = g - g^2$ and hence $g^{j-i}(1-g) = g(1-g)$ and thus $j - i = 1$, which is again a contradiction, as in the second case.

□