UNIVERSITÉ PARIS CITÉ

École doctorale Science Informatique de Paris Centre (ED386)
Institut de Recherche en Informatique Fondamental (UMR 8243)

Code-Based Post-Quantum Signature Schemes: from MPC-in-the-Head to Threshold Signatures

---

Par ELIANA CAROZZA

Thèse de doctorat de MATHEMATIQUES INFORMATIQUE

Dirigée par

IORDANIS KERENIDIS
and GEOFFROY COUTEAU and ANTOINE JOUX

Présentée et soutenue publiquement le 20/03/2025

Devant un jury composé de :

DAMIEN VERGNAUD	Professeur	Sorbonne Université, France	Rapporteur
GREG ZAVERUCHA	Software Engineer	Microsoft Engineer	Rapporteur
ALAIN COUVREUR	Directeur de Recherche	INRIA Saclay, France	Examinateur
EMMANUELA ORSINI	Assistant Professor	Bocconi University, Italy	Examinatrice
ADELINE ROUX-LANGLOIS	Directrice de Recherche	CNRS, France	Examinatrice
ANTOINE JOUX	Permanent Researcher	Saarbrücken University, Germany	Membre invité
GEOFFROY COUTEAU	Chargé de Recherche	CNRS, IRIF, Université Paris Cité	Directeur de thèse
IORDANIS KERENIDIS	Directeur de Recherche	CNRS, IRIF, Université Paris Cité	Directeur de thèse

Résumé

L'avènement des ordinateurs quantiques représente une menace majeure pour les systèmes cryptographiques classiques, comme l'a démontré l'algorithme de Shor, capable de résoudre efficacement le problème de factorisation des entiers. Cela a accéléré le développement de la cryptographie post-quantique, dédiée à concevoir des primitives sécurisées contre des adversaires quantiques. La cryptographie post-quantique se divise en plusieurs catégories selon les problèmes sur lesquels repose la sécurité des systèmes. Parmi les approches les plus prometteuses figurent les primitives basées sur les codes, dont la sécurité repose sur la complexité de problèmes comme le décodage de syndrome (SDP), intraitable pour les algorithmes classiques et quantiques.

Ce manuscrit définit et optimise des primitives cryptographiques basées sur les codes, en particulier des signatures numériques dérivées de preuves de connaissance à divulgation nulle via la transformation de Fiat-Shamir. Trois contributions principales sont présentées.

La première contribution introduit un protocole de preuve de connaissance à divulgation nulle en cinq tours basé sur le problème RSD, utilisant une technique avancée de conversion de témoin. Transformé en un schéma de signature numérique par l'heuristique de Fiat-Shamir, adaptée à la sécurité post-quantique, il bénéficie d'une optimisation par une nouvelle technique d'hypercube, atteignant des vitesses compétitives.

La deuxième contribution améliore l'efficacité du paradigme MPC-in-the-Head en définissant des fonctions pseudorandom puncturables multi-instances (PPRFs) conçues pour les protocoles basés sur les codes. En remplaçant les approches à base de hachage par des chiffrements en bloc à clé fixe, les PPRFs réduisent les temps de signature et de vérification jusqu'à 55×, établissant de nouvelles références de performance.

La troisième contribution étend le schéma initial en un cadre de signature seuil, relevant les défis de l'adaptation des schémas MPC pour une signature collaborative multi-utilisateur. La méthodologie minimise le compromis entre la taille des signatures et le nombre d'utilisateurs, surpassant les techniques naïves. Appliquée au schéma basé sur les codes, elle aboutit à une solution de signature seuil pratique et efficace.

En conclusion, ce manuscrit propose des avancées majeures en cryptographie post-quantique basée sur les codes, en relevant les défis de sécurité et d'efficacité. Alliant conception théorique rigoureuse et pertinence pratique, ces contributions s'alignent sur les objectifs du NIST et répondent aux besoins émergents tels que la blockchain et les systèmes décentralisés.

Mots-clés: Cryptographie à base de codes, Conception de primitives symétriques, Cryptographie post-quantique, Schémas de signature, MPC-in-the-Head, Signatures seuil, Signature post-quantique, Signature à base de codes, Décodage de syndrome régulier, Preuve à divulgation nulle de connaissance.

Abstract

The advent of quantum computers poses a critical threat to classical cryptographic systems, as demonstrated by Shor's algorithm, which efficiently solves the integer factorization problem. This has accelerated the development of post-quantum cryptography, a field dedicated to designing cryptographic primitives secure against quantum adversaries. Post-quantum cryptography is categorized into distinct areas based on the problems underpinning the security of its systems. Among the most promising approaches are code-based primitives, whose security relies on the complexity of coding theory problems such as the syndrome decoding problem (SDP), which remains intractable for both classical and quantum algorithms.

This manuscript focuses on defining and optimizing code-based cryptographic primitives, with a particular emphasis on digital signatures derived through the Fiat-Shamir transformation of zero-knowledge proofs of knowledge. It presents three main contributions.

The first contribution introduces a five-round zero-knowledge proof of knowledge protocol based on the RSD problem, employing an advanced witness conversion technique. The protocol is then transformed into a digital signature scheme using the Fiat-Shamir heuristic, adapted for post-quantum security. Performance is further optimized through a novel hypercube technique, achieving competitive speeds relative to existing schemes.

The second contribution focuses on improving the efficiency of the MPC-in-the-Head paradigm by defining multi-instance puncturable pseudorandom functions (PPRFs) specifically designed for code-based protocols: replacing traditional hash-based approaches with fixed-key block ciphers, the proposed PPRFs reduce signing and verification times by up to 55×, establishing new performance benchmarks for MPC-in-the-Head-based schemes.

The third contribution extends the initial digital signature scheme into a threshold signature framework, addressing the challenges of efficiently adapting MPC-based schemes for multi-user collaborative signing. The proposed methodology minimizes the trade-off between signature size and user count, outperforming naive concatenation techniques. Applied to the earlier code-based signature scheme, this approach results in a practical and efficient threshold signature solution.

In conclusion, this manuscript presents significant advancements in code-based post-quantum cryptography by addressing fundamental challenges in security and efficiency. Combining rigorous theoretical design with practical relevance, the contributions align with the goals of the NIST standardization process and address emerging needs such as blockchain and decentralized systems.

Keywords: Code-based cryptography, Design of Symmetric Primitives, Post-Quantum Cryptography, Signature Schemes, MPC-in-the-Head, Threshold signatures,Post-quantum Signature, Code-based signature, Regular syndrome decoding, Zero-knowledge Proof.

Acknowledgments

These almost four years of PhD have been an intense mix of excitement, frustration, breakthroughs, and countless moments of existential crisis. It has been a journey that pushed me out of my comfort zone, made me see things from new perspectives, and occasionally left me wondering what on earth I had signed up for. But through it all, I have been incredibly lucky to have the support of amazing people who not only helped me grow as a researcher but also did their best to keep my sanity somewhat intact.

This section is my small way of saying thank you to those who guided me, challenged me, believed in me, and, most importantly, tolerated me through the ups and downs of this adventure!

My advisors

I don't think I can fully express how grateful I am to my advisor, Geoffroy. He has been, without question, the most important person in this whole journey. From the start, he has supported me in every possible way—not just academically, but on a personal level too: he never pressured me, never made me feel like I wasn't doing enough, always gave me the space to work at my own pace while making sure I knew he was there whenever I needed guidance. That kind of balance is rare, and I feel incredibly lucky to have had it.

Geoffroy is, above all, an exceptional person. He's always positive, always pushing forward, never discouraged by setbacks. No matter what, he finds a way to stay optimistic, and that attitude has helped me more times than I can count. There were moments when I doubted myself, when I struggled to see my own progress, and every single time, he was there reminding me of what I had accomplished, of what I was capable of. Having someone like that in my corner made all the difference.

From the academic side he's simply brilliant: he always has new ideas, new directions to explore, new ways to push research forward. His ability to turn concepts into concrete strategies, to see potential where others see dead ends, is something I've admired from day one. He made research exciting, dynamic, and full of possibilities, and working with him has shaped the way I approach problems, think critically, and navigate the world of cryptography. For all of this, I genuinely couldn't have wished for a better advisor.

I would also like to sincerely thank Antoine for his role in my academic path. Even though I never had the chance to spend a year in Germany as originally planned, our discussions during meetings have still played an important role in shaping parts of my research. His ideas and perspectives have contributed to the development of my work, particularly by inspiring new directions in my papers. While our collaboration was not as extensive as initially expected, his input during our exchanges has been valuable, and I appreciate the insights he shared throughout this journey.

My PhD thesis scientific committee
A huge thank you to the members of my PhD jury, Alain Couvreur, Emmanuela Orsini, Adeline Roux-Langlois and Iordanis Kerenidis for taking the time to evaluate my work. A special thanks to Damien Vergnaud and Greg Zaverucha for their thorough review and feedback. Knowing that experts of your caliber took the time to read and analyze my work in such depth is something I truly appreciate.
I also want to acknowledge the members of my CSI, Alain Couvreur and Jean-Pierre Tillich, who, year after year, provided feedback on my research and progress.

Collaborations That Made a Difference
Research is never a solitary endeavor, and I have been fortunate to collaborate with brilliant minds throughout this PhD. A huge thank you to my coauthors, Dung Bui, Geoffroy Couteau, Dahmun Goudarzi and Antoine Joux.

A special thanks to Matthieu Rivain and Thibault Feneuil, as well as the entire CryptoExperts team, for the time I spent with them during my internship. Working with such brilliant researchers was both challenging and rewarding, and the time spent at the company was an invaluable experience.

IRIF collegues
These past years at IRIF have been way more than just research—they've been about the energy and the countless moments that made this place feel like a second home. The daily office life, the completely random yet always interesting conversations, the shared frustrations over deadlines, the coffee breaks that somehow turned into long debates about literally anything... all of it is something I'm really going to miss.
A massive thank you to all my office colleagues, to the crypto group, and to everyone at IRIF who made this whole experience what it was. Working alongside such brilliant people has been both inspiring and a constant reminder of just how much there is to learn. This place, and the people in it, have defined my PhD experience in a way that's hard to put into words. It won't be the same without it, and I already know I'm going to miss it way more than I'd like to admit.

Sharing an office with a great group of people can make all the difference in a PhD, and I've been lucky to have Giulia, Adam, Sacha and Enzo as part of this experience. I'm grateful for the time we shared, the conversations we had, and the atmosphere we built together in our little corner of IRIF.

The person who turned this entire PhD experience into something that felt like home is Dung, my office mate who quickly became one of my best friends. It's tough to capture just how significant she's been—she's not only been part of my daily life, but also fundamentally influenced the person I've become over these four years. We've shared everything from the tiniest, silliest bits of office gossip to the big existential questions that come with work (and, well, life). She's been there for every laugh, every rant, and every last-minute scramble to meet a deadline, and at some point our desks practically merged into one shared existence. What sets her apart is this almost magical blend of a razor-sharp mind and a deeply genuine heart. She's funny in the best possible ways—sarcastic but always on point—and her support has been unwavering on the days I was convinced I'd never see the end of a project.

The thought that, in just a few weeks, she won't be next to me spinning around in her chair, ready to announce "Okay, you know what—" before launching into three hours of gossip, feels completely surreal. No one else has witnessed (or tolerated) my quirks, moods, and anxieties more closely than she has, day in and day out, for these intense four years. She's seen the good, the bad, and the downright ridiculous, yet never once made me feel undeserving of the friendship we share.
I'm beyond grateful that our paths crossed in such a personal and meaningful way.

Another person who made my daily life at the office so much brighter is Maël, an incredible colleague and friend —someone I could always count on to share a laugh, talk through a research idea, or just compare notes on life in general.
Maël's easygoing nature and positive attitude always helped lighten the mood, especially on those tougher days when stress levels soared. He's the kind of person you can have a spontaneous conversation with, and before you know it, you're both diving into in-depth discussions that somehow flow between music festivals, the big existential crises of life. , and the latest Netflix shows.
I'm grateful for the camaraderie we developed and the way he always managed to keep things both real and fun. Working alongside him made even the most routine office days more enjoyable, and I truly appreciate his friendship. Wherever our paths take us next, I hope we'll keep sharing stories, ideas, and laughs.

It would be impossible to look back on my time at IRIF without mentioning Ulysse, whose unique style and constant stream of witty commentary brought a spark of fun (and sometimes outright hilarity) to my office life every single day. His sense of humor was often the perfect antidote to long days of wrestling with proofs, and I'm convinced he could make even the most mundane task sound like an epic adventure. I could count on him for genuine friendship and honest advice. Even if we started out cracking jokes, we somehow always managed to end up sharing deeper thoughts about life. I'm truly grateful for the balance he brought: he never took himself—or anyone else—too seriously, yet he was always there when I needed real support.

From the very start of this PhD journey, Clément has been there: we started this adventure together, and along the way, we've crossed paths in different cities, sat through countless talks at conferences, and navigated the chaos of research side by side. I'm grateful for all the moments we've shared.

My friends and family

To my lifelong friends and to my family—those who have been with me since I can remember, and those who have joined the adventure more recently—you probably already know how grateful I am to have you in my life. While I could fill pages with the depth of my feelings, I'll do my best to maintain a bit of formality here and avoid turning these acknowledgments into an overly personal, tear-filled confession (like what happened with my Master's thesis).

My deepest thanks go to my entire family: my grandparents, aunts, uncles, and cousins, who have been a constant source of warmth and encouragement throughout this journey. Each of you, in your own way, has played a role in shaping who I am, and your unwavering support means more than I can ever express. I hope you know this accomplishment isn't just mine, but ours together.

To Mom and Dad: I know that watching me live so far from home isn't easy, especially when all you really want is for me to be under your roof again, just like when I was little. Yet, despite the heartache and distance, you have never stopped believing in me or taking pride in every step I've taken along the way. It's only now, as I look back on the countless hours you spent teaching me, supporting me, and pushing me to be my best, that I realize just how much you both sacrificed—sometimes in ways I may never fully comprehend.
You gave me the freedom to chase my dreams, even when it meant you wouldn't get to see me every day or even every month. You cheered me on from afar, always reminding me that love has no boundaries, no matter how many miles separate us. When things got tough, I'd hear your voices in my head, telling me to keep going, to be strong, to remember that home is always there if I need to come back.
Thank you for showing me, in countless ways, that I could trust my instincts and chart my own path. Every success I achieve is as much yours as it is mine, because I couldn't have done any of this without the unwavering foundation you built for me. You are, and will always be, the reason I had the courage to leave in the first place.
I hope you know that each time I face a new challenge or celebrate a new milestone, you're right there with me in my thoughts. Your belief in me has given me the strength to keep going when I doubted myself, and knowing I'm still your child—no matter where life takes me—makes every bit of this journey worth it. I love you more than words could ever say.

To Giuseppe: I can't believe how quickly you're growing up. Watching you change, find your passions, and become your own person has been both thrilling and, if I'm honest, a bit surreal. It's as though every time I turn around, you've taken a giant leap forward.
Part of why I embarked on this journey was to show you that it's okay to dream big, to step outside your comfort zone, and to keep pushing forward even when things get tough. I hope that, if nothing else, you'll see that pursuing what you love—no matter how daunting—can lead you to places you never thought possible. You're already such a remarkable individual, and I have no doubt you'll do amazing things in life.
I'm proud of you in ways that are hard to put into words, and I'll always be here cheering you on, no matter where our paths lead. Keep going, keep growing, and don't ever be afraid to take the leap. You have so much talent, determination, and heart—use them all, and the world will open up for you.

To my friends here in Paris, thank you for making this city feel like home from the very start. Moving to a place where I knew no one could have been daunting, but with you, I never once felt alone. You surrounded me with so much warmth, laughter, and love that even in a city that was once unfamiliar, I always felt like I belonged. You turned everyday moments into cherished memories and proved that home is not just a place, but the people who make you feel safe, understood, and deeply cared for. I can't imagine this chapter of my life without you, and I am endlessly grateful for each and every one of you.

To my friends in Italy, I love that no matter where we are, we find a way to bridge the distance with a quick text, a silly meme, or a marathon video chat. Every time I come back home, it's like no time has passed: we pick up right where we left off, as if the miles between us never existed. I hope you realize how much I value each of you—your humor, your warmth, and the simple fact that you've chosen to be part of my life.

Contents

Résumé Substantiel en Français

CHAPTER 1

Vue d'ensemble de la Cryptographie Post-Quantique

L'éventuelle arrivée de l'informatique quantique a complètement transformé le monde de la cryptographie : les systèmes cryptographiques classiques tels que RSA, DSA et ECC, qui reposent sur des problèmes comme la factorisation d'entiers et les logarithmes discrets, sont confrontés à une vulnérabilité fondamentale. Ces problèmes, considérés comme intractables sur des machines classiques, peuvent être résolus efficacement à l'aide d'algorithmes quantiques comme l'algorithme de Shor [Sho97]. Par conséquent, les fondements de ces protocoles cryptographiques sont menacés, puisqu'un ordinateur quantique est capable de les résoudre en temps polynomial, rendant ainsi leur sécurité obsolète.

En réponse à ce défi, la cryptographie post-quantique (PQC) a émergé comme un domaine de recherche essentiel. L'objectif de la PQC est de développer des schémas cryptographiques qui restent sécurisés même face à des attaques quantiques, tout en maintenant leur robustesse dans des environnements classiques. Ce domaine explore diverses structures mathématiques censées résister aux calculs quantiques. Selon le problème mathématique sur lequel repose la sécurité, la cryptographie post-quantique se divise en :

- la cryptographie à base de treillis, avec des hypothèses notables telles que NTRU et Learning with Errors (LWE) ;
- la cryptographie multivariée (MQ), qui exploite la difficulté de la résolution de systèmes d'équations quadratiques sur des corps finis ;
- la cryptographie à base de fonctions de hachage, illustrée par les schémas de signature de Merkle ;
- la cryptographie à base de codes, incluant le cryptosystème de McEliece et les schémas reposant sur le problème de décodage de syndrome (SDP).

Parmi ces domaines, cette thèse se concentre sur la cryptographie à base de codes, un secteur bien établi de la sécurité post-quantique qui offre une combinaison prometteuse d'efficacité et de sécurité, grâce à la difficulté des problèmes sous-jacents et à la simplicité de leur structure algébrique par rapport à d'autres approches. Au cœur de ce domaine se trouve le *syndrome decoding problem (SDP)*, qui consiste à trouver un vecteur de faible poids de Hamming satisfaisant une équation linéaire définie par une matrice binaire. Sa difficulté est bien documentée et résiste tant aux algorithmes classiques que quantiques [Pra62; FJR22; Ste94a; CDM+22; BJM+12; Ber10; KT17; BJM+12; FS09], en faisant un pilier de la cryptographie moderne à base de codes.

Motivation pour la Cryptographie à Base de Codes

L'orientation vers la cryptographie à base de codes en cryptographie post-quantique est motivée par plusieurs raisons. Il ne s'agit pas seulement de la solidité de ses garanties de sécurité, mais aussi de son utilité pratique dans des systèmes cryptographiques réels : les schémas à base de codes sont étudiés depuis des décennies, et leur sécurité est solidement établie, reposant sur des problèmes mathématiques difficiles comme le décodage de syndrome. Ils offrent un équilibre solide entre rigueur théorique et implémentation pratique, ce qui en fait une option fiable pour construire des systèmes sécurisés dans un monde post-quantique, particulièrement lorsque la sécurité à long terme est une priorité.

Les facteurs les plus importants à prendre en compte lors du choix de la cryptographie à base de codes sont :

1. Fondement de sécurité établi : les problèmes cryptographiques à base de codes, en particulier le problème de décodage de syndrome, sont étudiés depuis plus de quarante ans. La proposition originelle du cryptosystème de McEliece en 1978 [McE78] a marqué le début d'un corpus de recherches solide.

2. Diversification et bénéfices pratiques : même si la cryptographie à base de codes est antérieure à la cryptographie à base de treillis, cette dernière a gagné beaucoup d'attention au cours des dernières décennies, de nombreux protocoles ayant été développés. Cependant, comme on peut facilement l'imaginer, s'appuyer sur une seule hypothèse est risqué en cas d'attaques inattendues : diversifier les primitives, comme l'a également encouragé le NIST, est crucial pour une meilleure sécurité. Parmi toutes les possibilités, la cryptographie à base de codes se distingue grâce à des constructions algébriques beaucoup plus simples —surtout sur $\mathbb{F}_2$—, la rendant plus difficile à attaquer via des méthodes algébriques complexes et, en même temps, plus facile à implémenter sur des dispositifs de base, ce qui la rend à la fois pratique et polyvalente.

3. Taille de signature efficace : historiquement, les schémas cryptographiques à base de codes ont été associés à de grandes clefs publiques, mais ils ont connu des améliorations notables en termes de taille de signature. Des avancées récentes [FJR22] ont donné

lieu à des schémas de signature plus compacts et efficaces, faisant des signatures à base de codes une option solide pour un déploiement réel dans des environnements post-quantiques.

Le *Syndrome Decoding Problem (SDP)* peut être décrit comme suit : étant donnée une matrice binaire aléatoire $H \in \mathbb{F}_2^{k \times K}$ et un vecteur cible $y \in \mathbb{F}_2^k$, l'objectif est de trouver un vecteur $x \in \mathbb{F}_2^K$ de poids de Hamming w tel que $H \cdot x = y$. La difficulté de calcul de ce problème constitue un élément clé pour construire des systèmes cryptographiques sécurisés, en particulier des schémas de signature post-quantiques [FJR22]. En particulier, l'application du problème de décodage de syndrome à la construction de signatures numériques a été explorée via différentes méthodes.

Signatures Numériques dans un Monde Post-Quantique

Les signatures numériques sont l'une des primitives cryptographiques les plus fondamentales, fournissant des garanties d'authenticité, d'intégrité et de non-répudiation. Dans un monde post-quantique, la construction de schémas de signature sécurisés représente un défi majeur, car les ordinateurs quantiques menacent directement les schémas de signature classiques, comme ceux basés sur la cryptographie à courbes elliptiques (ECC), étant donné que les ordinateurs quantiques peuvent résoudre efficacement les problèmes mathématiques sous-jacents à ces schémas classiques, tels que le logarithme discret. Cela motive la nécessité de signatures numériques post-quantiques qui restent sécurisées même face à des adversaires quantiques.

Les signatures basées sur le problème de décodage de syndrome offrent une solution prometteuse. En s'appuyant sur l'intractabilité du SDP, ces signatures sont intrinsèquement résistantes aux attaques quantiques. Une technique bien connue pour construire des signatures numériques sécurisées est l'heuristique de Fiat-Shamir [FS87], qui transforme les protocoles interactifs à jetons publics, tels que les preuves à divulgation nulle de connaissance, en schémas de signature non interactifs. Cette transformation remplace le défi aléatoire du vérificateur par un hachage déterministe du message, permettant ainsi de construire des signatures sans nécessiter d'interaction entre le prouveur et le vérificateur.

L'utilisation de la transformation de Fiat-Shamir a connu un succès particulier en cryptographie post-quantique. Dans le cadre de ce travail, cette transformation convertit une preuve à divulgation nulle de connaissance en 5 tours basée sur le problème de décodage de syndrome en un schéma de signature sûr et efficace.

L'un des principaux défis dans la conception de signatures post-quantiques est de trouver un équilibre entre sécurité, efficacité et taille de la signature. Les premiers schémas, tels que ceux dérivés de la preuve à divulgation nulle de Stern pour le décodage de syndrome [Ste94b], souffraient de tailles de signatures importantes dues à la nécessité de multiples répétitions pour réduire l'erreur de solidité. Pour atteindre une erreur de solidité de 2^{-128}, le protocole

de Stern requiert un nombre significatif de répétitions, ce qui entraîne des surcoûts de communication élevés.

Des avancées récentes [FJR22; KKW18; ZCD+20], notamment grâce à l'utilisation de techniques de calcul multipartite (MPC) telles que MPC-in-the-Head (MPCitH), ont résolu ces problèmes en réduisant la complexité de communication de la preuve tout en maintenant de fortes garanties de sécurité. Les schémas de signature basés sur MPCitH obtiennent des performances compétitives tant en termes de sécurité que d'efficacité, en faisant une option viable pour les signatures post-quantiques. Dans cette thèse, la conception et l'implémentation de schémas de signature efficaces à base de codes sont étudiées, en mettant l'accent sur l'utilisation de fonctions pseudorandomes puncturables (PPRF) et l'optimisation des protocoles MPC sous-jacents pour minimiser la taille des signatures et le coût de calcul.

Preuves à Divulgation Nulle de Connaissance (Zero-Knowledge Proofs)

Les preuves à divulgation nulle de connaissance (ZKPs) sont des protocoles cryptographiques qui permettent à un prouveur de démontrer qu'il possède une solution à un problème, sans révéler la solution elle-même. Une preuve à divulgation nulle de connaissance doit satisfaire trois propriétés clés :

- complétude : garantit qu'un prouveur honnête peut toujours convaincre le vérificateur de la véracité de l'énoncé ;
- solidité : assure qu'aucun prouveur malhonnête ne peut convaincre le vérificateur d'un énoncé faux avec une probabilité non négligeable ;
- divulgation nulle de connaissance : garantit que le vérificateur n'apprend rien au-delà de la validité de l'énoncé.

Dans le domaine de la cryptographie à base de codes, la première preuve à divulgation nulle de connaissance pour le problème de décodage de syndrome a été introduite par Stern [Ste94b]. Ce protocole permet à un prouveur de convaincre un vérificateur qu'il possède un témoin valide —c'est-à-dire une solution au problème de décodage de syndrome— sans le révéler. Malgré son caractère novateur, le protocole de Stern souffre d'une forte erreur de solidité : un prouveur malhonnête peut convaincre avec succès le vérificateur avec une probabilité de $2/3$ sans posséder la solution correcte. Pour atteindre une erreur de solidité de 2^{-128}, le protocole de Stern doit être répété de nombreuses fois, ce qui accroît considérablement le coût de communication. Par exemple, utiliser le protocole de Stern pour signer un seul message avec 128 tours engendrerait un coût de communication dépassant 1 MB, car chaque tour requiert la transmission de multiples engagements et réponses [Ste94a;

DFG+14]. À titre de comparaison, les schémas modernes comme Falcon atteignent des niveaux de sécurité similaires avec des tailles de signature d'environ 666 octets au niveau de sécurité NIST 1 et 1 280 octets au niveau 5 [FGP+18]. Cette différence notable illustre l'efficacité des schémas modernes en termes de coûts de communication comparés au protocole de Stern.

Le Rôle du Paradigme MPC-in-the-Head

4.4 joue un rôle déterminant dans la construction de protocoles cryptographiques post-quantiques. Initialement introduit par Ishai, Kushilevitz, Ostrovsky et Sahai [IKO+07], MPCitH permet au prouveur de simuler l'exécution d'un protocole de calcul multipartite (MPC) dans sa tête, en s'engageant sur les vues de plusieurs parties virtuelles participant au calcul. Le prouveur peut ensuite révéler sélectivement certaines parties de ces vues au vérificateur, ce qui permet à ce dernier de vérifier l'exactitude du calcul sans interaction directe de toutes les parties impliquées.

MPCitH s'est avéré particulièrement efficace dans la construction de signatures numériques post-quantiques : en s'engageant sur l'intégralité du calcul dans sa tête, le prouveur peut produire une preuve compacte de l'exactitude, nécessitant bien moins de communication que les protocoles à divulgation nulle de connaissance traditionnels. Cette réduction de la complexité de communication est d'autant plus cruciale dans un cadre post-quantique, où la taille des clefs publiques et des signatures pose souvent des défis pratiques.

Cette approche a été encore améliorée en combinant MPCitH avec des fonctions pseudo-randomes puncturables (PPRF), comme le soulignent Katz, Kolesnikov et Wang [KKW18], démontrant comment ces techniques peuvent conduire à des preuves à divulgation nulle de connaissance non interactives plus performantes. Les PPRF permettent au prouveur de révéler sélectivement certaines parties du calcul tout en gardant d'autres parties cachées. Cette révélation sélective est essentielle pour réduire le surcoût de communication associé aux preuves à divulgation nulle, car elle permet au prouveur d'ouvrir seulement les portions nécessaires du calcul pour la vérification. La combinaison de MPCitH et des PPRF a conduit au développement de schémas de signature post-quantiques très efficaces, offrant un compromis entre sécurité, efficacité et taille de la signature [FJR22].

Le présent manuscrit explore l'utilisation de MPCitH comme fondement pour construire des schémas de signature numérique post-quantiques efficaces, en optimisant les protocoles MPC sous-jacents et en intégrant des PPRF, permettant ainsi de réaliser des schémas de signature à la fois sûrs contre les adversaires quantiques et pratiques pour une utilisation réelle. La conception de ces schémas répond aux défis clés de la scalabilité, de l'efficacité de communication et de la sécurité, en faisant une option solide pour des applications cryptographiques post-quantiques.

Contributions de cette Thèse

Cette thèse apporte plusieurs contributions à la cryptographie post-quantique, en se concentrant sur les schémas de signature numérique construits à partir du problème de décodage de syndrome et du paradigme MPC-in-the-Head (MPCitH). Le travail présenté ici est le fruit de mes recherches de doctorat, au cours desquelles j'ai publié trois articles, chacun formant la base de l'un des chapitres principaux de la thèse.

Voici les principales contributions de la thèse :

- Deux nouveaux schémas de signature post-quantiques : la première avancée majeure est le développement de deux nouveaux schémas de signature basés sur le problème de décodage de syndrome classique. Les deux schémas utilisent le paradigme MPCitH pour combiner une sécurité élevée avec une efficacité pratique. Le premier schéma, expliqué au Chapitre 3, a été présenté dans [CCJ23], un article co-rédigé avec Geoffroy Couteau et Antoine Joux, présenté à Eurocrypt23. Il introduit une nouvelle manière d'utiliser MPCitH avec le décodage de syndrome pour obtenir des signatures compactes et réduire les coûts de communication. Le second schéma, construit sur la base du premier, améliore encore les performances et corrige certaines limites de la conception initiale. Il constitue une partie de [BCC+24], un article co-rédigé avec Dung Bui, Geoffroy Couteau, Dahmun Goudarzi et Antoine Joux, présenté à Asiacrypt24.

- Fonction pseudorandome puncturable (PPRF) améliorée : le Chapitre 4 se concentre sur l'amélioration de l'efficacité des signatures basées sur MPCitH, comme présenté dans [BCC+24]. Ce travail optimise les PPRF pour obtenir une signature plus rapide et réduire la communication, aspects cruciaux pour la mise en pratique de ces schémas dans des scénarios réels. Ces améliorations démontrent l'impact significatif de la nouvelle conception sur l'utilisabilité de toutes les signatures reposant sur MPCitH.

- Schéma de signature de seuil : la contribution finale [CC24], décrite au Chapitre 5, étend les travaux des signatures individuelles aux signatures de seuil. Cela permet à plusieurs utilisateurs de signer un message collectivement de manière sécurisée et distribuée, ce qui est particulièrement utile dans des applications nécessitant une confiance partagée. Ce travail est une extension naturelle des schémas précédents et fait actuellement l'objet d'une soumission.

Ces contributions répondent à certains des plus grands défis de la conception de schémas de signature post-quantiques, notamment en matière de passage à l'échelle, d'efficacité et de sécurité. Chaque chapitre s'appuie sur le précédent, illustrant comment la recherche a progressivement évolué pendant mon doctorat.

Mes Articles

[BCC+24] Dung Bui, Eliana Carozza, Geoffroy Couteau, Dahmun Goudarzi, and Antoine Joux. “Short Signatures from Regular Syndrome Decoding, Revisited”. In: *ASIACRYPT 2024 (to appear)*. 2024. URL: https://eprint.iacr.org/2024/252.

[CC24] Eliana Carozza and Geoffroy Couteau. *On Threshold Signatures from MPC-in-the-Head*. Cryptology ePrint Archive, Paper 2024/1897. 2024. URL: https://eprint.iacr.org/2024/1897.

[CCJ23] Eliana Carozza, Geoffroy Couteau, and Antoine Joux. “Short Signatures from Regular Syndrome Decoding in the Head”. In: *EUROCRYPT 2023, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. LNCS. Springer, Heidelberg, Apr. 2023, pp. 532–563. DOI: 10.1007/978-3-031-30589-4_19.

CHAPTER 2

Introduction

Overview of Post-Quantum Cryptography

The possible advent of quantum computing has completely changed the world of cryptography: classical cryptographic systems like RSA, DSA, and ECC, which rely on problems such as integer factorization and discrete logarithms, face a fundamental vulnerability. These problems, considered computationally intractable for classical machines, can be efficiently solved using quantum algorithms like Shor's algorithm [Sho97]. As a result, the foundations of these cryptographic protocols are under threat, as quantum computers can solve them in polynomial time, rendering their security obsolete.

In response to this challenge, post-quantum cryptography (PQC) has emerged as a critical area of study. The goal of PQC is to develop cryptographic schemes that remain secure even against quantum attacks while maintaining their robustness in classical environments. This field explores various mathematical structures that are supposed to be resistant to quantum computations. Depending on the mathematical problem on which security relies, post-quantum cryptography can be divided into:

- Lattice-based cryptography, with notable assumptions like NTRU and Learning with Errors (LWE);
- Multivariate quadratic (MQ) cryptography, which exploits the difficulty of solving systems of quadratic equations over finite fields;
- Hash-based cryptography, exemplified by Merkle signature schemes;
- Code-based cryptography, including the McEliece cryptosystem and schemes relying on the syndrome decoding problem (SDP).

Among these, the thesis focuses on code-based cryptography, a well-established area in post-quantum security that offers a promising combination of efficiency and security, due

to the hardness of the underlying problems and the simplicity of their algebraic structure compared to other approaches. Central in this field is the syndrome decoding problem (SDP), which involves finding a vector with a small Hamming weight that satisfies a linear equation defined by a binary matrix. Its hardness has been well-documented, resisting both classical and quantum algorithms [Pra62; FJR22; Ste94a; CDM+22; BJM+12; Ber10; KT17; BJM+12; FS09], making it a cornerstone of modern code-based cryptography.

Motivation for Code-Based Cryptography

The focus on code-based cryptography in post-quantum cryptography is motivated by several reasons. It is not just about its strong security guarantees but also its practical use in real cryptographic systems: code-based schemes have been studied for decades, and their security is well-established, relying on hard mathematical problems like syndrome decoding. They offer a solid balance between theoretical rigor and practical implementation and this makes them a reliable option for building secure systems in a post-quantum world, especially when long-term security is a priority. The most important factors to consider when choosing code-based cryptography are:

1. Established Security Foundation: code-based cryptographic problems, particularly the syndrome decoding problem, have been extensively studied for over four decades. The original proposal of the McEliece cryptosystem in 1978 [McE78] marked the beginning of a well-established body of research.

2. Diversification and Practical Benefits: even if code-based cryptography is chronologically older than Lattice-based cryptography, the latter has gained a lot of attention in the last decades, with many protocols being developed. However, as one can easily imagine, relying on just one assumption is risky in case of unexpected attacks: diversifying primitives, as NIST has also encouraged, is crucial for better security. Among all the possibilities, code-based cryptography stands out because it uses much simpler algebraic constructions –especially over $\mathbb{F}_2$–, making it harder to attack with complex algebraic methods and at the same time, easier to implement on basic devices, which makes it both practical and versatile.

3. Efficient Signature Size: code-based cryptographic schemes, while historically associated with large public keys, have seen significant improvements in terms of signature size. Recent advancements [FJR22], have led to more compact and efficient signature schemes, making code-based signatures a strong candidate for real-world deployment in post-quantum environments.

The *Syndrome Decoding Problem (SDP)* can be described as follows: given a random binary matrix $H \in \mathbb{F}_2^{k \times K}$ and a target vector $y \in \mathbb{F}_2^k$, the objective is to find a vector $x \in \mathbb{F}_2^K$ of Hamming weight w such that $H \cdot x = y$. The computational hardness of this problem is

a key component in constructing secure cryptographic systems, particularly post-quantum digital signature schemes [FJR22]. In particular, the application of the syndrome decoding problem in constructing digital signatures has been explored through various methodologies.

Digital Signatures in a Post-Quantum World

Digital signatures are one of the most fundamental cryptographic primitives, providing assurances of authenticity, integrity, and non-repudiation. In the post-quantum world, building secure digital signature schemes is an important challenge, as quantum computers pose a direct threat to classical signature schemes, such as those based on elliptic curve cryptography (ECC), since quantum computers can efficiently solve the mathematical problems that underlie these classical schemes, such as discrete logarithms. This motivates the need for post-quantum digital signatures that remain secure even in the presence of quantum adversaries.

Signatures based on the syndrome decoding problem, offer a promising solution. By leveraging the intractability of SDP, these signatures are inherently resistant to quantum attacks. A well-known technique for constructing secure digital signatures is through the Fiat-Shamir heuristic [FS87], which transforms interactive public-coin protocols, such as zero-knowledge proofs of knowledge, into non-interactive signature schemes. This transformation replaces the verifier's random challenge with a deterministic hash of the message, allowing for the construction of signatures without requiring interaction between the prover and the verifier. The use of the Fiat-Shamir transformation has been particularly successful in post-quantum cryptography. In the context of this work, the transformation converts a 5-round zero-knowledge proof of knowledge based on the syndrome decoding problem into a secure and efficient signature scheme.

One of the key challenges in designing post-quantum signatures is balancing security, efficiency, and signature size. Early schemes, such as those derived from Stern's zero-knowledge proof for syndrome decoding [Ste94b], suffered from large signature sizes due to the need for multiple repetitions to reduce the soundness error. To achieve a soundness error of 2^{-128}, Stern's protocol required a significant number of repetitions, resulting in substantial communication overheads.

Recent advancements [FJR22; KKW18; ZCD+20], particularly through the use of multi-party computation (MPC) techniques such as MPC-in-the-Head (MPCitH), have addressed these issues by reducing the communication complexity of the proof while maintaining strong security guarantees. MPCitH-based signature schemes achieve competitive performance in terms of both security and efficiency, making them a viable option for post-quantum digital signatures. In this thesis, the design and implementation of efficient code-based signature schemes are explored, with a focus on utilizing puncturable pseudorandom functions (PPRFs) and optimizing the underlying MPC protocols to minimize signature size and computational cost.

Zero-Knowledge Proofs

Zero-knowledge proofs (ZKPs) are cryptographic protocols that allow a prover to demonstrate knowledge of a solution to a problem without revealing the solution itself. A zero-knowledge proof must satisfy three key properties:

- completeness: ensures that an honest prover can always convince the verifier of the truth of the statement;
- soundness: guarantees that no dishonest prover can convince the verifier of a false statement with non-negligible probability;
- zero-knowledge: ensures that the verifier learns nothing beyond the validity of the statement.

In the domain of code-based cryptography, the first zero-knowledge proof of knowledge for the syndrome decoding problem was introduced by Stern [Ste94b]. This protocol enables a prover to convince a verifier that they possess a valid witness –i.e., a solution to the syndrome decoding problem– without revealing it. Despite its innovation, Stern's protocol suffers from a high soundness error: a dishonest prover can successfully convince the verifier with a probability of $2/3$ without possessing the correct solution. To achieve a soundness error of 2^{-128}, Stern's protocol must be repeated multiple times, leading to a significant increase in communication overhead. For instance, using Stern's protocol to sign a single message with 128 rounds would result in a communication cost exceeding 1 MB, as each round requires transmitting multiple commitments and responses [Ste94a; DFG+14]. In contrast, modern schemes like Falcon achieve comparable levels of security with signature sizes of approximately 666 bytes at NIST security level 1 and 1,280 bytes at level 5 [FGP+18]. This stark difference highlights the efficiency of modern schemes in terms of communication overhead compared to Stern's protocol.

The Role of the MPC-in-the-Head Paradigm

The MPC-in-the-Head (MPCitH) 4.4 paradigm is a critical innovation in the construction of post-quantum cryptographic protocols. Originally introduced by Ishai, Kushilevitz, Ostrovsky, and Sahai [IKO+07], MPCitH allows the prover to simulate the execution of a multi-party computation (MPC) protocol in his head, committing to the views of several virtual parties that participate in the computation. The prover can then selectively reveal certain parts of these views to the verifier, enabling the verifier to check the correctness of the computation without the need for direct interaction between all parties involved.

MPCitH has proven to be particularly effective in the construction of post-quantum digital signatures: by committing to the entire computation in his head, the prover can

produce a compact proof of correctness that requires significantly less communication than traditional zero-knowledge protocols. This reduction in communication complexity is especially important in post-quantum settings, where the size of public keys and signatures often poses practical challenges.

This approach was further enhanced by combining MPCitH with puncturable pseudorandom functions (PPRFs), as highlighted in the work of Katz, Kolesnikov, and Wang [KKW18], which demonstrated how these techniques can lead to improved non-interactive zero-knowledge proofs. PPRFs allow the prover to selectively reveal certain parts of the computation while keeping other parts hidden. This selective revelation is critical in reducing the communication overhead associated with zero-knowledge proofs, as it enables the prover to open only the necessary portions of the computation for verification. The combination of MPCitH and PPRFs has led to the development of highly efficient post-quantum signature schemes, which offer a balance between security, efficiency, and signature size [FJR22].

This manuscript explores the use of MPCitH as the foundation for constructing efficient post-quantum digital signature schemes, by optimizing the underlying MPC protocols and integrating PPRFs, and making it possible to achieve signature schemes that are both secure against quantum adversaries and practical for deployment in real-world cryptographic systems. The design of these schemes addresses the key challenges of scalability, communication efficiency, and security, making them a strong candidate for post-quantum cryptographic applications.

Contributions of this Thesis

This thesis makes several contributions to post-quantum cryptography, focusing on digital signature schemes built on the syndrome decoding problem and the MPC-in-the-Head (MPCitH) paradigm. The work presented here is the result of my PhD research, during which I published three articles, each of which forms the basis for one of the main chapters of the thesis.

Here are the main contributions of the thesis:

- Two New Post-Quantum Signature Schemes: the first big step was the development of two new signature schemes based on the regular syndrome decoding problem. Both schemes use the MPCitH paradigm to combine strong security with practical efficiency. The first scheme, explained in Chapter 3, was introduced in [CCJ23], a paper co-authored with Geoffroy Couteau and Antoine Joux, presented at Eurocrypt23. It introduced a new way of using MPCitH with syndrome decoding to achieve compact signatures and reduce communication costs. The second scheme, built on the first one, improves the performance further and addresses some of the limitations of the initial design. It represents a part of [BCC+24], a paper co-authored with Dung Bui, Geoffroy Couteau, Dahmun Goudarzi, and Antoine Joux, presented at Asiacrypt24.

- Improved Puncturable Pseudorandom Function (PPRF): Chapter 4 focuses on improving the efficiency of MPCitH-based signatures, as presented in [BCC+24]. The work enhances PPRFs to achieve faster signing and reduced communication overhead, both of which are critical for the practicality of these schemes in real-world scenarios. These improvements demonstrate the significant impact of the new design on the usability of all MPCitH-based signatures.
- Threshold Signature Scheme: the final contribution [CC24], described in Chapter 5, extends the work from individual signatures to threshold signatures. This allows multiple users to sign a message together in a secure and distributed way, which is particularly useful in applications that require collective trust. This work is a natural extension of the previous schemes and it is currently under submission.

These contributions address some of the biggest challenges in designing post-quantum signature schemes, including scalability, efficiency, and security. Each chapter builds on the previous one, showing how the research evolved step by step throughout my PhD.

My Papers

[BCC+24] Dung Bui, Eliana Carozza, Geoffroy Couteau, Dahmun Goudarzi, and Antoine Joux. "Short Signatures from Regular Syndrome Decoding, Revisited". In: *ASIACRYPT 2024 (to appear)*. 2024. URL: https://eprint.iacr.org/2024/252.

[CC24] Eliana Carozza and Geoffroy Couteau. *On Threshold Signatures from MPC-in-the-Head*. Cryptology ePrint Archive, Paper 2024/1897. 2024. URL: https://eprint.iacr.org/2024/1897.

[CCJ23] Eliana Carozza, Geoffroy Couteau, and Antoine Joux. "Short Signatures from Regular Syndrome Decoding in the Head". In: *EUROCRYPT 2023, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. LNCS. Springer, Heidelberg, Apr. 2023, pp. 532–563. DOI: 10.1007/978-3-031-30589-4_19.

CHAPTER 3

Preliminaries

This chapter provides the necessary background for the research presented in this thesis. It starts with the basic mathematical notation used throughout the manuscript to ensure clarity and consistency. Then other key concepts are described, including computational and security assumptions, probability models, and cryptographic primitives that form the basis for the thesis. The last part of this chapter is dedicated to code-based cryptography, especially the syndrome decoding problem and its regular variant which are at the core of the presented works.

Contents

3.1 Mathematical Notation

Given a set S, $s \leftarrow\!\$\ S$ indicates that s is uniformly sampled from S. Given an integer $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \cdots, n\}$. For bitstrings, $\{0,1\}^k$ denotes the set of all bitstrings of length k, and $\{0,1\}^*$ denotes the set of all bitstrings of any length.

Modulo. Let $x \in \mathbb{Z}$ and $k > 0$ be a positive integer. The reduction of x modulo k, denoted as $x \mod k$, is the remainder of the division of x by k. The set $\mathbb{Z}_k = \{0, 1, \cdots, k-1\}$ represents all integers modulo k, with addition and multiplication defined modulo k. Based on this, the ring of integers modulo k is denoted as $(\mathbb{Z}_k, +, \cdot)$, and the group of invertible integers modulo k is denoted as $(\mathbb{Z}_k^*, \cdot)$. For a general k, the size of this group is given by Euler's totient function $\varphi(k)$, *i.e.* the number of integers n such that $1 \leq n \leq k$ and $\gcd(k, n) = 1$, where $\gcd$ is the greatest common divisor. If k is a prime number, $\mathbb{Z}_k$ forms a field, and $\mathbb{Z}_k^* = \mathbb{Z}_k \setminus \{0\}$. Equality modulo k is expressed as $a = b \mod k$.

Vectors and matrices. Bold lowercase is used for vectors and uppercase for matrices. The notation $A||B$ denotes the horizontal concatenation of matrices A, B, and $A//B$ denotes their vertical concatenation. The symbol Id_n denotes the $n \times n$ identity matrix. By default, vectors are always considered as columns. Given a vector $\mathbf{v}$, it is often written as $\mathbf{v} = (\mathbf{v}_1, \cdots, \mathbf{v}_n)$ to indicate that $\mathbf{v}$ is a (vertical) concatenation of n subvectors $\mathbf{v}_i$. This slight abuse of notation avoids the (more precise, but cumbersome) notation $\mathbf{v} = (\mathbf{v}_1^\intercal, \cdots, \mathbf{v}_n^\intercal)^\intercal$. Given $\mathbf{u}, \mathbf{v} \in \{0,1\}^n$, $\mathbf{u} \oplus \mathbf{v}$ denotes the bitwise-XOR of $\mathbf{u}$ and $\mathbf{v}$, and $\mathsf{HW}(\mathbf{u})$ denotes the Hamming weight of $\mathbf{u}$, i.e., its number of nonzero entries.

Compact and expanded forms. Given two integers n, m and an index $i \in [n]$, $\mathbf{e}_i \in \mathbb{F}_2^n$ denotes the length-n unit $\mathbb{F}_2$-vector whose i-th entry is 1. Given w indices $(i_1, \cdots, i_w) \in [n]^w$, the previous notation is extended to $\mathbf{e_i} = (\mathbf{e}_{i_1}, \cdots, \mathbf{e}_{i_w})$, the concatenation of w unit vectors. Noise vectors are typically manipulated in *compact form*, *i.e.*, as elements $(i_1, \cdots, i_w)$ of $[m]^w$, where each entry $i_j \in [m]$ indicates the position of the 1 in the j-th length-bs unit vector. The mapping Expand is used to denote the transformation which, given a noise vector $x = (x_1, \cdots, x_w) \in [m]^w$, outputs the vector $\mathbf{e}_x = (\mathbf{e}_{x_1}, \cdots, \mathbf{e}_{x_w}) \in \mathbb{F}_2^K$. The vector $\mathbf{e}_x$ is called the *expanded form* of x.

Permutations. The set $\mathsf{Perm}(n)$ denotes all permutations of $[n]$. In this manuscript, permutations over $[n]$ are typically used to shuffle the entries of a length-n vector or to shuffle the *blocks* of a vector which is the concatenation of n blocks. For example, given two integers n, m, a vector $\mathbf{v} \in [m]^n$ and a permutation $\pi : [n] \mapsto [n]$, $\pi(\mathbf{v})$ denotes the vector $(v_{\pi(1)}, v_{\pi(2)}, \cdots, v_{\pi(n)})$. If $\mathbf{v}$ is the concatenation of n subvectors $(\mathbf{v}_1, \cdots, \mathbf{v}_n)$, then $\pi(\mathbf{v})$ denotes the vector $(\mathbf{v}_{\pi(1)}, \cdots, \mathbf{v}_{\pi(n)})$.

Code parameters. A linear code of length n and dimension k over a finite field $\mathbb{F}_q$ is defined as a k-dimensional subspace of $\mathbb{F}_q^n$.

Definition 3.1.1. *For a linear code C over a finite field, a parity-check matrix H is defined such that a vector $\mathbf{c}$ is a codeword in C if and only if $Hc^T = 0$. In other words, H is a matrix whose null space corresponds to the code C. The rows of H represent the coefficients of the parity-check equations that define the code.*

Throughout this manuscript, K denotes the number of columns in the parity-check matrix H, and k denotes the number of its rows. Equivalently, K is the codeword length, and $K - k$ is the dimension of the code. The weight of the noise is denoted by w, which always divides K. The block size is denoted by $\mathsf{bs} \leftarrow K/w$ and a w-regular noise vector is sampled as a concatenation of w random unit vectors (the *blocks*) of length bs. Reg_w denotes the set of all length-K w-regular vectors.

Cyclic shifts. Given a vector $\mathbf{u} \in \mathbb{F}_2^n$ and $i \in [n]$, $\mathbf{u} \downarrow i$ is used to denote the vector $\mathbf{u}$ cyclically shifted by i steps (in other words, $\mathbf{u} \downarrow i$ is the convolution of $\mathbf{u}$ and $\mathbf{e}_i$). The notation $\mathsf{Shift}(\mathbf{u}, i)$ is also used to denote $\mathbf{u} \downarrow i$. The concept is extended to a *block-by-block* cyclic shift of vectors: given a vector $\mathbf{u} \in \mathbb{F}_2^K$, viewed as a sequence of blocks $(\mathbf{u}_1, \cdots, \mathbf{u}_m) \in \mathbb{F}_2^{K/m} \times \cdots \times \mathbb{F}_2^{K/m}$, and a vector of shifts $x \in [l]^m$, $\mathbf{u} \downarrow x$ denotes the vector obtained by shifting the blocks of $\mathbf{u}$ according to x. Specifically, $\mathbf{u} \downarrow x = (\mathbf{v}_1, \cdots, \mathbf{v}_m)$ where each $\mathbf{v}_i$ is the vector obtained by cyclically shifting (downward) the vector $\mathbf{u}_i$ by x_i steps. To ensure clarity, $\downarrow$ is treated as a "top priority" operator: by default, for any other operation op, $\mathbf{u} \downarrow x \ \mathsf{op}\ \mathbf{v}$ is interpreted as $(\mathbf{u} \downarrow x) \ \mathsf{op}\ \mathbf{v}$ and not $\mathbf{u} \downarrow (x \ \mathsf{op}\ \mathbf{v})$.

Binary tree. For a tree of size 2^D, and each leaf $i \in [2^D]$, $\mathsf{CoPath}(i)$ denotes the set of all the siblings of the nodes in the path from the root to the i-th leaf. If bit-decompose i is expressed as $\sum_{j=1}^{D} 2^{j-1} \cdot i_j$ for $i_j \in \{0, 1\}$, the associated value of the i-th leaf is defined as $X_i := X_{i_1,\ldots,i_D}$.

Pòlya's Enumeration Theorem. Pòlya's Enumeration Theorem is a powerful combinatorial tool used to count distinct configurations under the action of a symmetry group. Let G be a finite group acting on a finite set X, and let C be a set of colors. The number of distinct colorings of X, up to the action of G, is given by:

$$Z(G, C) = \frac{1}{|G|} \sum_{g \in G} |C|^{\mathrm{Fix}(g)},$$

where $|G|$ is the order of the group G, and $\mathrm{Fix}(g)$ denotes the number of elements in X that remain fixed under the action of g.

3.2 Algorithmic and Computational Models

Turing Machine. A Turing machine is a mathematical model used to describe computations. It processes input on a tape, reading symbols and performing operations such as writing, moving left or right, or halting based on a set of predefined rules. A formal definition is below:

Definition 3.2.1. *A Turing machine is a 7-tuple $M = (Q, \Gamma, b, \Sigma, \delta, q_0, F)$, where Q is the set of states, Γ is the tape alphabet, $b \in \Gamma$ is the blank symbol, $\Sigma \subseteq \Gamma \setminus \{b\}$ is the input on the tape, δ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states.*

Since the machine halts when it reaches a state in F, this model can be useful for defining the computations that an algorithm can perform. In this manuscript, computations are modeled using probabilistic Turing machines, which are Turing machines that can make use of randomness via a uniform random tape. Given a probabilistic Turing machine $\mathcal{A}$ and an input x, $y \leftarrow\!\!\$\ \mathcal{A}(x)$ –or $y \leftarrow \mathcal{A}(x; r)$ when the random coins are explicit– is written to indicate that y is sampled by running $\mathcal{A}$ on x with a uniform random tape.

Interactive protocols. In interactive protocols, where more parties are involved, each party can be modeled as an interactive probabilistic Turing machine. These are multi-tape Turing machines equipped with additional tapes for communication: one read-only and one write-only. Interaction between machines is modeled by allowing them to exchange messages through their communication tapes, enabling the simulation of interactive protocols between parties.

Multi-Party Computation (MPC). Multi-party computation is a particular class of interactive protocols where n multiple parties, each modeled as an interactive probabilistic Turing machine, aim to jointly compute a function $f(x_1, x_2, \ldots, x_n)$ of their private inputs $x_1, x_2, \cdots, x_n$, without revealing any additional information about their inputs: the interaction between parties is modeled as message exchanges over their communication tapes. An MPC protocol is considered secure if, for every adversary corrupting a subset of the parties, there exists a simulator that can replicate the view of the adversary using only the inputs and outputs that the adversary is allowed to observe. This guarantees that the protocol leaks no information about honest parties' inputs beyond what is revealed by the output of the function itself.

Polynomial-Time Algorithms. A probabilistic algorithm is referred to as a *PPT algorithm* (Probabilistic Polynomial-Time algorithm) if it runs in time polynomial in the size of its input, for all inputs and all random coins.

Communication Cost. The total communication cost of a protocol $\mathcal{P}$ is denoted by $\mathsf{Comm}(\mathcal{P})$, defined as the sum of all messages exchanged between parties during the execution of $\mathcal{P}$. For a protocol with n rounds, where each round i involves message exchanges m_i^j of length $|m_i^j|$ from a party P_j, the communication cost is given by $\mathsf{Comm}(\mathcal{P}) = \sum_{i=1}^{n} \sum_j |m_i^j|$. When applied to multi-party protocols, $\mathsf{Comm}(\mathcal{P})$ is considered as the aggregate communication over all parties. For example, in the case of $\mathbb{F}_2^\lambda$ vectors communicated over t rounds in an MPC protocol, $\mathsf{Comm}(\mathcal{P})$ is computed as the total bit-length of all transmitted vectors.

Preprocessing Phase. The preprocessing phase of a protocol $\mathcal{P}$ includes all the steps that the parties can perform before knowing their private inputs. In the case of Multi-Party Computation (MPC), the preprocessing phase can be:

- *Independent Preprocessing Phase*: this involves tasks that don't even depend on the function f that the parties want to compute. Examples include generating correlated randomness or cryptographic keys that can be reused across different protocols.
- *Dependent Preprocessing Phase*: here, the steps in the preprocessing are tailored to the specific function f: the parties want to generate special correlated data or specific preprocessing material required for securely evaluating f.

In scenarios involving a trusted external party, preprocessing may include distributing pre-computed correlated randomness or secret shares to participants. This can significantly reduce the computational and communication effort required by the parties during the main execution of the protocol. The preprocessing phase aims to offload as much work as possible to a stage where the private inputs are not yet known so that the online phase —when inputs are available—can be as efficient as possible. Its complexity is measured by the total time and communication required to complete all preprocessing steps.

Runtime. The runtime of an algorithm or protocol $\mathcal{A}$ is the total time required to complete its execution, typically expressed as a function of the input size n. For deterministic algorithms, runtime is represented by the number of basic operations performed, denoted as $T(n)$. For randomized or probabilistic algorithms, runtime may include an expectation over possible random choices. In cryptographic protocols, runtime includes both computation time and any delaying time that is necessary for communication rounds. For example, in an MPC protocol, runtime includes the combined time for local computations by each party and the time for message exchanges during different rounds. Runtime analysis is crucial for evaluating the efficiency of $\mathcal{A}$, especially in practical implementations.

3.3 Principles of Security and Probability

Negligiblity. A function f is said to be *negligible*, denoted $f(x) = \text{negl}(x)$, if for every constant $c \in \mathbb{N}$, there exists an $x_0 \in \mathbb{N}$ such that for all $x \geq x_0$, it holds that $\|f(x)\| \leq 1/x^c$. A function f is said to be *overwhelming* if $1 - f$ is negligible.

Security parameter. In cryptography, a *security parameter*, denoted by λ in this manuscript, is used to quantify the computational difficulty of breaking a system: all efficient algorithms are assumed to run in polynomial time in λ. Usually, all the parameters in a designed protocol are chosen such that it guarantees λ bit of security, implying that the best-known attack requires 2^λ steps to succeed. In this manuscript, as often in cryptography, the chosen value for λ is 128, since 2^{128} computational steps are considered infeasible with current technology.

Statistical indistinguishability. The statistical distance between two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ over a finite set S is defined as

$$\sum_{i \in S} \left| \Pr_{x \leftarrow \mathcal{D}_0}[x = i] - \Pr_{x \leftarrow \mathcal{D}_1}[x = i] \right|.$$

Definition 3.3.1. *Two distributions are said to be statistically indistinguishable if their statistical distance is negligible. This implies that no algorithm, even one with unbounded computational power, has more than a negligible advantage in distinguishing between the two distributions.*

Independent Random Variables. A set of random variables $X_1, \ldots, X_n$ is independent if the joint probability $\Pr[X_1 = x_1, \ldots, X_n = x_n]$ equals the product $\prod_{i=1}^{n} \Pr[X_i = x_i]$. Independence ensures that variables do not provide any information about others, which is essential for analyzing statistical indistinguishability between distributions derived from independent sources.

Markov bound. Let X be a non-negative random variable with expected value $\mu = \mathbb{E}[X]$. For any $a > 0$, the Markov Bound states that:

$$\Pr(X \geq a) \leq \frac{\mathbb{E}[X]}{a}.$$

This inequality provides an upper bound on the probability that X takes a value significantly larger than its expected value.

Chernoff bound. Let $X_1, X_2, \ldots, X_n$ be n independent random variables, where each $X_i \in \{0, 1\}$, and let $X = \sum_{i=1}^{n} X_i$ denote their sum. If $\mu = \mathbb{E}[X]$ is the expected value of X, then for any $\delta > 0$:

- The probability that X exceeds $(1 + \delta)\mu$ is bounded as:

$$\Pr[X \geq (1 + \delta)\mu] \leq \left(\frac{e^{\delta}}{(1 + \delta)^{1+\delta}} \right)^{\mu}.$$

- The probability that X is less than $(1 - \delta)\mu$ is bounded as:

$$\Pr[X \leq (1 - \delta)\mu] \leq \left(\frac{e^{-\delta}}{(1 - \delta)^{1-\delta}} \right)^{\mu}.$$

These bounds indicate that the probability of X deviating significantly from its expected value decreases exponentially as the deviation increases. For this reason, The Chernoff Bound is widely used to analyze the concentration of sums of independent random variables around their expectation.

Chernoff Bound for Hypergeometric Distributions. Let X be a random variable following a hypergeometric distribution, where N is the population size, K is the number of successes in the population, and n is the number of samples drawn without replacement. The expected value of X is given by

$$\mu = \mathbb{E}[X] = n \cdot \frac{K}{N}.$$

For any $\delta > 0$, the following bounds hold:

- The probability that X exceeds $(1 + \delta)\mu$ is bounded as:

$$\Pr[X \geq (1 + \delta)\mu] \leq \exp\left(-\frac{\delta^2 \mu}{2(1 + \delta/3)} \right).$$

- The probability that X is less than $(1 - \delta)\mu$ is bounded as:

$$\Pr[X \leq (1 - \delta)\mu] \leq \exp\left(-\frac{\delta^2 \mu}{2} \right).$$

Stirling's Inequality. The Stirling inequality provides bounds for factorials, which are particularly useful in probabilistic and combinatorial analyses. For any positive integer n, the factorial $n!$ satisfies:

$$\sqrt{2\pi n} \left(\frac{n}{e} \right)^n \leq n! \leq \sqrt{2\pi n} \left(\frac{n}{e} \right)^n e^{\frac{1}{12n}}.$$

This inequality is essential when approximating probabilities involving large factorials, such as in binomial or hypergeometric distributions. In cryptographic applications where asymptotic bounds play a crucial role, this inequality offers a precise way to estimate large combinatorial terms.

Birthday Paradox. The Birthday Paradox is a counterintuitive probability result usually applied in the analysis of hash functions and collision probabilities. This result shows that in a set of n uniformly chosen samples from a space of size N, the probability of at least one collision (*i.e.* two samples being identical) becomes significant even for $n \ll N$. More precisely, the probability of observing a collision can be approximated as:

$$\Pr[\text{collision}] \approx 1 - e^{-n(n-1)/(2N)}.$$

For $n = O(\sqrt{N})$, this probability becomes non-negligible, with a collision probability exceeding $1/2$ for $n \geq 1.17\sqrt{N}$. This phenomenon has significant implications in cryptography, especially for hash functions: if the hash function's output space is of size 2^N, the expected number of samples required to observe a collision is approximately $2^{N/2}$.

3.4 Cryptographic Primitives

3.4.1 Hash function

Hash functions are cryptographic primitives that take inputs of any length and produce fixed-size outputs. They represent a core component in cryptography: since their main role is to provide compact and secure representations of data, they are widely used in applications like commitment schemes and digital signatures –hashing the message first allows signing a smaller, fixed-size digest instead of the whole message, which improves efficiency while preserving security–.

Definition 3.4.1. *A cryptographic hash function is a deterministic algorithm $\mathcal{H} : \{0,1\}^* \to \{0,1\}^\ell$, mapping inputs of arbitrary size to fixed-length outputs, that satisfy the following properties:*

- ***Pre-image Resistance**: Given $y \in \{0,1\}^\ell$, it should be computationally infeasible to find any $x \in \{0,1\}^*$ such that $\mathcal{H}(x) = y$.*
- ***Second Pre-image Resistance**: Given $x \in \{0,1\}^*$, it should be computationally infeasible to find $x' \neq x$ such that $\mathcal{H}(x) = \mathcal{H}(x')$.*
- ***Collision Resistance**: It should be computationally infeasible to find two distinct inputs $x, x' \in \{0,1\}^*$ such that $\mathcal{H}(x) = \mathcal{H}(x')$.*

Modern hash functions, such as SHA-2 [Nat15a] and SHA-3 [Nat15b], are designed to satisfy these properties while maintaining computational efficiency. The security of these functions is often analyzed in the *Random Oracle Model* (see section 3.4.7.1) [BR93], where they are seen as truly random functions. However, for a hash function with an output length of ℓ bits, the security level depends on the specific property:

- For pre-image resistance, the security level is 2^{ℓ}, as finding a pre-image requires brute-forcing over the entire output space.
- For collision resistance, the security level is reduced to $2^{\ell/2}$ due to the *Birthday Paradox*.

Applications in Cryptography. Hash functions play an important role in many cryptographic protocols:

- Ensuring message integrity and authenticity, as in Message Authentication Codes (MACs) like HMAC [KBC97].
- Enhancing efficiency in digital signatures by compressing the data to be signed, while maintaining security [BR96].
- Securing password storage, often combined with salting techniques to protect against pre-image attacks.
- Enabling commitment schemes by providing computationally binding and hiding commitments [CD01].
- Powering proof-of-work systems, such as in blockchain technologies like Bitcoin [Nak08].

Hash Functions in Post-Quantum Cryptography. The advent of quantum computing presents new challenges for traditional hash functions. Grover's algorithm [Gro96] enables quantum computers to search an unsorted database in $O(2^{n/2})$ time, effectively halving the security level of classical hash functions for pre-image resistance. For example, a hash function providing 256-bit classical security would offer only 128 bits of quantum security against pre-image attacks. Different, and more complex, is the case for collision resistance, where the security is already $2^{n/2}$ even in the classical setting due to the Birthday Paradox. However, the introduced vulnerability necessitates adapting hash functions for post-quantum cryptography.

The NIST post-quantum cryptography standardization process [Natng] has highlighted hash-based schemes as promising candidates for quantum-resistant cryptographic protocols. Several approaches have been proposed:

- Increase Output Size: by doubling the output length of classical hash functions, the security level can be restored to match pre-quantum standards.

- Hash-Based Cryptography: cryptographic schemes such as Merkle Trees [Mer89] and SPHINCS+ rely on the security of hash functions. These schemes are well-suited for post-quantum applications since they rely on minimal cryptographic assumptions.
- Structured Designs: new hash functions based on stronger mathematical constructs like lattices (e.g., Learning With Errors) introduce additional hardness assumptions, enhancing their security against quantum attacks.

In conclusion, hash functions remain a cornerstone of cryptography: through strategic adaptations and new innovative designs, they can address the challenges of a post-quantum world while staying versatile tools with a really wide range of applications.

Salted hash function. A *salt* is a random value appended to an input before hashing. Represented as $H(x\|s)$, where s is the salt and $\|$ denotes concatenation, the purpose of salting is to ensure that even identical inputs produce distinct hash values, thus enhancing security. Salted hash functions are widely adopted in systems where unique hash values are necessary, such as in password storage and authentication systems [BHO+12].

Typically, salts are random and of a fixed length (often 128 bits or more), providing sufficient uniqueness across multiple hashing operations. Although the salt does not need to be secret, it should be stored securely alongside the hash output to maintain consistency for verification purposes [KSW+97]. Salting also finds applications in cryptographic protocols, adding an extra layer of uniqueness in multi-target attack scenarios [BHO+12].

3.4.2 Commitment Schemes

Commitment schemes are one of the most fundamental primitives in cryptography. They allow a committer to commit to a value s, producing a commitment c, which can later be opened to reveal s to the verifier. Such schemes ensure that the committed value cannot be changed after the commitment phase (binding), while hiding the committed value during the commitment phase (hiding). This section provides a formal definition of a commitment scheme in the plain model. However, within the manuscript, the commitment scheme in the Random Oracle Model (ROM) will also be implicitly utilized, though it will not be formally detailed

Definition 3.4.2 (Commitment Scheme)**.** *A commitment scheme* Π *is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{Comm}, \mathsf{Open})$ *satisfying the following:*

- $\mathsf{Setup}(1^\lambda)$*: Generates the public parameters,* i.e. *the message space* $\mathcal{M}$*, the commitment space* $\mathcal{C}$*, the opening space* $\mathcal{D}$*, and the randomness space* $\mathcal{R}$*.*
- $\mathsf{Comm}(m; r)$*: Given a message* $m \in \mathcal{M}$ *and randomness* $r \in \mathcal{R}$*, outputs a commitment-opening pair* $(c, d) \in \mathcal{C} \times \mathcal{D}$*.*

- $\mathsf{Open}(c, d, m)$: *Verifies the validity of the opening* (m, d) *for the commitment* c *and outputs a bit* $b \in \{0, 1\}$.

To be considered secure a commitment scheme must satisfy correctness, hiding, and binding.

Definition 3.4.3. *A commitment scheme is correct if for all public parameters* $\mathcal{M}, \mathcal{C}, \mathcal{R}, \mathcal{D} \leftarrow \mathsf{Setup}(1^\lambda)$*, any message* $m \in \mathcal{M}$*, and any randomness* $r \in \mathcal{R}$*, it holds that* $\mathsf{Open}(c, d, m) = 1$ *for* $(c, d) \leftarrow \mathsf{Comm}(m; r)$.

Definition 3.4.4. *A commitment scheme is hiding if for all PPT adversaries* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{\mathsf{hiding}}_{\mathcal{A}}(\lambda)$ *is negligible, where the advantage is defined as*

$$\mathsf{Adv}^{\mathsf{hiding}}_{\mathcal{A}}(\lambda) = \left|\Pr[\mathcal{A}(c_0) = 1] - \Pr[\mathcal{A}(c_1) = 1]\right|,$$

and $c_b \leftarrow \mathsf{Comm}(m_b; r)$ *for* $b \in \{0, 1\}$*,* $m_0, m_1 \in \mathcal{M}$*, and uniformly random* $r \in \mathcal{R}$.

Definition 3.4.5. *A commitment scheme is binding if for all PPT adversaries* $\mathcal{A}$*, the success probability* $\mathsf{Succ}^{\mathsf{binding}}_{\mathcal{A}}(\lambda)$ *is negligible, where*

$$\mathsf{Succ}^{\mathsf{binding}}_{\mathcal{A}}(\lambda) = \Pr[\mathsf{Open}(c, d, m) = 1 \wedge \mathsf{Open}(c, d', m') = 1 \wedge m \neq m'].$$

3.4.3 Pseudorandom Generator (PRG)

A pseudorandom generator (PRG) is a cryptographic primitive, designed to take a short, truly random seed and expand it into a much longer sequence such that it appears to be computationally indistinguishable from a uniformly random string to any efficient adversary. This property represents the base for the security of PRG-based cryptographic schemes [BM82; Yao82]. The concept of pseudorandomness was formalized by Blum and Micali [Blu82], and later extended by Yao [Yao08] to define the security of PRGs based on computational hardness assumptions.

Definition 3.4.6. *A pseudorandom generator (*PRG*) is a deterministic polynomial-time algorithm* $G : \{0, 1\}^s \to \{0, 1\}^m$ *that takes as input a* sd $x \in \{0, 1\}^\lambda$ *of length* λ *and outputs a string* $G(x) \in \{0, 1\}^m$ *of length* $m > \lambda$*, where* $m = m(\lambda)$ *is a polynomial function of* λ.
It satisfies the pseudorandomness property: for any probabilistic polynomial-time algorithms $\mathcal{A}$*, there exists a negligible function* $\mathit{negl}(\cdot)$ *such that, for* $x \leftarrow \{0, 1\}^\lambda$ *and* $y \leftarrow\!\!\$\ \{0, 1\}^m$

$$\left|\Pr[\mathcal{A}(G(x)) = 1] - \Pr[\mathcal{A}(y) = 1]\right| \leq \mathit{negl}(\lambda)$$

where $x \leftarrow \{0, 1\}^\lambda$ *and* $y \leftarrow \{0, 1\}^m$ *is uniformly random.*

Pseudorandom generators (PRGs) rely on computational hardness assumptions, such as the difficulty of factoring integers or solving the discrete logarithm problem. These assumptions guarantee that it is computationally infeasible to invert the generation process: such property is fundamental to the security of PRGs, both in classical cryptography and in post-quantum frameworks [Reg06; Pei10].

Definition 3.4.7. *Let $G : \{0,1\}^s \to \{0,1\}^n$ be a* PRG*, and let $\mathcal{D}$ be a distinguisher, an algorithm that attempts to differentiate between the output of G and a uniformly random sequence. The advantage of $\mathcal{D}$ is defined as:*

$$Adv_G(\mathcal{D}) = \left|\Pr[\mathcal{D}(G(U_s)) = 1] - \Pr[\mathcal{D}(U_n) = 1]\right|,$$

where U_s is a uniformly random seed of length s, and U_n is a uniformly random sequence of length n.
A PRG is (t, ϵ)-secure if, for any distinguisher $\mathcal{D}$ running in time t, the advantage $Adv_G(\mathcal{D}) \leq \epsilon$.

The (t, ϵ) security model allows quantifying the practical security of PRGs, as it provides a concrete bound on the computational resources required by an adversary to break the PRG's pseudorandomness. This model is widely used in cryptographic proofs, as it enables designers to analyze the security of PRGs within the constraints of real-world adversaries. For instance, a PRG with $(2^{80}, 2^{-40})$-security is considered resistant to adversaries with resources equivalent to 2^{80} operations, achieving a distinguishing probability of at most 2^{-40} [BR96]. Hence, (t, ϵ)-secure PRGs are essential in scenarios where the generated pseudorandom sequences must meet stringent security requirements, such as in-stream ciphers and cryptographic protocols where even slight statistical biases could lead to vulnerabilities. This security notion also allows for a practical trade-off between efficiency and security; by adjusting t and ϵ, cryptographic schemes can be tailored to specific security levels based on the capabilities of adversaries.

3.4.4 Pseudorandom Functions (PRF)

The concept of a Pseudorandom Function (PRF) plays a foundational role in cryptographic applications, providing a deterministic yet indistinguishable mapping from inputs to outputs. Formally, a PRF is defined as follows:

Definition 3.4.8 (Pseudorandom Function)**.** *A pseudorandom function (PRF) is a deterministic function $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, where $\mathcal{K}$ denotes the key space, $\mathcal{X}$ the input space, and $\mathcal{Y}$ the output space.* F *satisfies the following property: for any PPT adversary $\mathcal{A}$, given oracle access to either F_k (for a random key k) or a truly random function $f : \mathcal{X} \to \mathcal{Y}$, the adversary's advantage in distinguishing the two is negligible:*

$$Adv^{\mathsf{prf}}_{\mathcal{A}}(\lambda) = \left|\Pr[Exp^{\mathsf{prf}-0}_{\mathcal{A}}(\lambda) = 1] - \Pr[Exp^{\mathsf{prf}-1}_{\mathcal{A}}(\lambda) = 1]\right| \leq negl(\lambda).$$

where the experiments are defined in Figure 3.1.

Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{prf}-b}(\lambda)$:

1. If $b = 0$: the challenger samples $k \leftarrow\!\!\$\ \mathcal{K}$ and provides oracle access to F_k to the adversary $\mathcal{A}$.
2. If $b = 1$: the challenger provides oracle access to a truly random function $f : \mathcal{X} \to \mathcal{Y}$ to the adversary $\mathcal{A}$.
3. The adversary $\mathcal{A}$ interacts with the oracle and eventually outputs a bit b'.
4. The experiment outputs 1 if $b' = b$, and 0 otherwise.

Figure 3.1: Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{prf}-b}(\lambda)$ *for pseudorandom functions.*

3.4.4.1 Constructing a PRF Using a PRG: the GGM Construction The construction of a pseudorandom function (PRF) from a pseudorandom generator (PRG) was introduced by Goldreich, Goldwasser, and Micali in their seminal 1986 paper *How to Construct Random Functions* [GGM86]. This construction, commonly referred to as the *GGM Construction*, demonstrates how a deterministic function that expands a short random seed can be used to create a function indistinguishable from a truly random one.

At its core, the GGM Construction relies on building a binary tree structure using the PRG to define a PRF. Here's the high-level idea:

- Start with a short seed k (the key of the PRF) and treat it as the root of the tree.
- Use the PRG G to expand the seed into two outputs: one corresponding to the left child (0) and the other to the right child (1).
- For a given input $x \in \{0,1\}^d$, interpret x as a sequence of directions in the tree, where each bit of x determines whether to move left or right at each level.
- The leaf reached by following the path defined by x is the output $\mathsf{F}_k(x)$ of the PRF.

The security of this construction relies on the pseudorandomness of G. If G is secure, then the adversary cannot distinguish the output of the PRF from that of a truly random function, even with oracle access. A formal definition is provided below.

Definition 3.4.9. *Let* $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ *be a pseudorandom generator that expands a* λ*-bit seed into a* 2λ*-bit pseudorandom output. The PRF* $\mathsf{F} : \{0,1\}^\lambda \times \{0,1\}^d \to \{0,1\}^\lambda$ *is defined recursively as follows: Given a PRG* G*, the PRF* F *is constructed as:*

- *Interpret the input* $x \in \{0,1\}^d$ *as a binary string* $x_1, x_2, \ldots, x_d$*.*

- *Start with $v_0 = k$, where k is the PRF key.*
- *At each level i, compute $v_i = \mathsf{G}(v_{i-1})$ and split v_i into two halves: $v_i[0]$ and $v_i[1]$.*
- *Follow the path determined by x: at level i, select $v_i[x_i]$.*
- *The final output $\mathsf{F}_k(x)$ is the value at the leaf node.*

This construction ensures that F inherits the pseudorandomness of G, assuming that G is secure: the security of the PRF can be shown via a hybrid argument, where the adversary's ability to distinguish F_k from a truly random function is reduced to breaking the pseudorandomness of G.

3.4.5 Puncturable Pseudorandom Functions (PPRF)

A puncturable pseudorandom function (PPRF), introduced in [KPT+13; BW13; BGI14], is a type of pseudorandom function that allows selective omission of specific output values, known as punctures, while preserving pseudorandomness for other input values: given an input x, and a PRF key k, a *punctured* key, denoted $k\{x\}$ allows evaluating F at every point except for x, and does not reveal any information about the value $F.\mathsf{Eval}(k, x)$.

By allowing punctures, PPRFs enable flexible, fine-grained control over which outputs are revealed, adding a layer of security and functionality to cryptographic protocols: this is the reason why they are particularly useful in cryptographic applications that require selective revocation of function values, such as in zero-knowledge proofs, secure multiparty computation, and digital signatures [BGI16a; GGH+13]. For example, the seminal GGM PPRF [GGM86] is a PPRF that has been used in multiple contexts to compress many seeds into a short seed, such that one can succinctly open all-but-one seeds. In particular, it was first used in the context of MPC-in-the-head signatures in [KKW18].

Definition 3.4.10. *A puncturable pseudorandom function F is a family of pseudorandom functions $\{F_k : \{0,1\}^n \to \{0,1\}^m\}_{k \in \mathcal{K}}$ indexed by a key $k \in \mathcal{K}$ with the following properties:*

- ***Pseudorandomness**: for all PPT algorithms $\mathcal{A}$, there exists a negligible function $\mathit{negl}(\cdot)$ such that*
$$|\Pr[\mathcal{A}(F_k(x)) = 1] - \Pr[\mathcal{A}(r) = 1]| \leq \mathit{negl}(n),$$
where $k \leftarrow \mathcal{K}$, $x \leftarrow \{0,1\}^n$, and $r \leftarrow \{0,1\}^m$ is uniformly random.
- ***Puncturability**: For any input $x^* \in \{0,1\}^n$, there exists an efficient puncturing algorithm $\mathit{Puncture}(k, x^*)$ that outputs a modified key k_{x^*}, which allows evaluation of F_k at all points $x \neq x^*$ but hides the value $F_k(x^*)$.*

- ***Security under Puncturing***: *for all PPT algorithms $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that*

$$|\Pr[\mathcal{A}(F_k(x)|k_{x^*}) = 1] - \Pr[\mathcal{A}(r) = 1]| \leq negl(n),$$

where $k \leftarrow \mathcal{K}$, $x^* \in \{0,1\}^n$, $k_{x^*} = Puncture(k, x^*)$, and $r \leftarrow \{0,1\}^m$ is uniformly random.

In the previous section 3.4.4.1, a PRF was constructed from a PRG using the GGM construction, which organizes the computation as a binary tree, leveraging a length-doubling pseudorandom generator G. Hence, the resulting PRF is puncturable, as its tree structure allows specific inputs to be excluded by removing certain sub-tree.

A PPRF can be constructed in various ways, depending on how the underlying PRG G is instantiated: of course, the instantiation directly affects the efficiency and security of the resulting PPRF. Below, two common approaches to defining G are discussed, based on widely used cryptographic primitives: symmetric encryption and hash functions.

AES-Based PPRFs. Using the Advanced Encryption Standard (AES) [01] as the core primitive, the PRG can be instantiated as:

$$\mathsf{G}(x) = (\mathsf{AES}_{k_0}(x) \oplus x, \mathsf{AES}_{k_1}(x) \oplus x),$$

where k_0 and k_1 are keys corresponding to the left and right branches. This construction ensures that each branch label is uniquely determined by the node label x and its corresponding key. Additionally, these constructions leverage the efficiency and security properties of AES, ensuring that the outputs remain indistinguishable from random for anyone without knowledge of $k_{0,1}$. AES-based PPRFs are particularly suitable for hardware-optimized implementations, where their performance advantages are significant.

Hash-Based PPRFs. Alternatively, G can be instantiated using a cryptographic hash function H, such as SHA-256 [Nat15a]. In this case, G can be defined as:

$$\mathsf{G}(x) = (H(x\|k_0), H(x\|k_1)),$$

where k_0, and k_1 are keys for the left and right branches again, and $\|$ denotes concatenation. While hash-based PPRFs are not as optimized for hardware as AES-based constructions, they offer greater flexibility in software environments. This makes them ideal for applications where simplicity and compatibility are prioritized, such as secure multiparty computation (MPC) and zero-knowledge protocols.

3.4.6 Pseudorandom correlation generators

The notion of pseudorandom correlation generator (PCG) is recalled from [BCG+19b]. At a high level, a PCG for some target ideal correlation takes as input a pair of short, correlated seeds and outputs long correlated pseudorandom strings, where the expansion procedure is deterministic and can be applied locally. The definitions below are taken directly from [BCG+20b].

Definition 3.4.11 (Correlation generator)**.** *A PPT algorithm* C *is referred to as a* correlation generator *if, on input* 1^λ*,* C *outputs a pair of elements in* $\{0,1\}^n \times \{0,1\}^n$ *for* $n \in \mathsf{poly}(\lambda)(\lambda)$.

The security definition of PCGs requires the target correlation to satisfy a technical requirement, which states that it must be possible to efficiently sample from the conditional distribution of $\mathcal{R}_0$ given $\mathcal{R}_1 = r_1$ and vice versa. This property holds for the correlations considered here.

Definition 3.4.12 (Reverse-sampleable correlation generator)**.** *Let* C *be a correlation generator.* C *is said to be* reverse sampleable *if there exists a PPT algorithm* $\mathsf{RSample}$ *such that, for* $\sigma \in \{0,1\}$*, the correlation obtained via:*

$$\{(\mathcal{R}'_0, \mathcal{R}'_1) \mid (\mathcal{R}_0, \mathcal{R}_1) \leftarrow\$ \mathsf{C}(1^\lambda), \mathcal{R}'_\sigma := \mathcal{R}_\sigma, \mathcal{R}'_{1-\sigma} \leftarrow\$ \mathsf{RSample}(\sigma, \mathcal{R}_\sigma)\}$$

is computationally indistinguishable from $\mathsf{C}(1^\lambda)$.

Definition 3.4.13 (Pseudorandom Correlation Generator (PCG))**.** *Let* C *be a reverse-sampleable correlation generator. A* pseudorandom correlation generator (PCG) for C *is a pair of algorithms* $(\mathsf{PCG.Setup}, \mathsf{PCG.Expand})$ *with the following syntax:*

- $\mathsf{PCG.Setup}(1^\lambda)$ *is a PPT algorithm that, given a security parameter* λ*, outputs a pair of seeds* (c_0, c_1)*;*
- $\mathsf{PCG.Expand}(\sigma, c_\sigma)$ *is a polynomial-time algorithm that, given a party index* $\sigma \in \{0,1\}$ *and a seed* c_σ*, outputs a bit string* $\mathcal{R}_\sigma \in \{0,1\}^n$.

The algorithms $(\mathsf{PCG.Setup}, \mathsf{PCG.Expand})$ *satisfy the following:*

- **Correctness.** *The correlation obtained via:*

  $$\{(\mathcal{R}_0, \mathcal{R}_1) \mid (c_0, c_1) \leftarrow\$ \mathsf{PCG.Setup}(1^\lambda), R_\sigma \leftarrow \mathsf{PCG.Expand}(\sigma, c_\sigma) \text{ for } \sigma \in \{0,1\}\}$$

  is computationally indistinguishable from $\mathsf{C}(1^\lambda)$.

- **Security.** *For any $\sigma \in \{0,1\}$, the following two distributions are computationally indistinguishable:*

$$\{(c_{1-\sigma}, \mathcal{R}_\sigma) \mid (c_0, c_1) \leftarrow\!\!\$\ \mathsf{PCG.Setup}(1^\lambda), \mathcal{R}_\sigma \leftarrow \mathsf{PCG.Expand}(\sigma, c_\sigma)\} \text{ and}$$
$$\{(c_{1-\sigma}, \mathcal{R}_\sigma) \mid (c_0, c_1) \leftarrow\!\!\$\ \mathsf{PCG.Setup}(1^\lambda), \mathcal{R}_{1-\sigma} \leftarrow \mathsf{PCG.Expand}(\sigma, c_{1-\sigma}), R_\sigma \leftarrow\!\!\$\ \mathsf{RSample}(\sigma, \mathcal{R}_{1-\sigma})\}$$

where RSample *is the reverse sampling algorithm for correlation* C.

It is noted that PCG.Setup could simply output a sample from C. To avoid this trivial construction, the seed size is required to be significantly shorter than the output size.

3.4.6.1 Programmable PCGs. A programmable PCG allows for the generation of multiple PCG keys such that part of the correlation generated remains consistent across different instances. Programmable PCGs are utilized to construct n-party correlated randomness from the 2-party correlated randomness generated by the PCG. When expanding n-party shares (e.g., Beaver triples) into a sum of 2-party shares, programmable PCGs ensure consistent pseudorandom values across these cross terms. The formal definition is as follows:

Definition 3.4.14. *A tuple of algorithms* $\mathsf{PCG} = (\mathsf{PCG.Setup}, \mathsf{PCG.Expand})$ *following the syntax of a standard* PCG*, but where* $\mathsf{PCG.Setup}(1^\lambda)$ *takes additional random inputs $\rho_0, \rho_1 \in \{0,1\}^*$, is referred to as a* programmable PCG *for a simple bilinear 2-party correlation C_e^n (specified by $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$) if the following conditions hold:*

- ***Correctness.*** *The correlation obtained via:*

$$\left\{ ((R_0, S_0), (R_1, S_1)) \middle| \begin{array}{l} \rho_0, \rho_1 \xleftarrow{\$} \$, (k_0, k_1) \leftarrow\!\!\$\ \mathsf{PCG.Setup}(1^\lambda, \rho_0, \rho_1), \\ (R_\sigma, S_\sigma) \leftarrow \mathsf{PCG.Expand}(\sigma, k_\sigma) \text{ for } \sigma \in \{0,1\} \end{array} \right\}$$

is computationally indistinguishable from $C_e^n(1^\lambda)$.

- ***Programmability.*** *There exists a public efficiently computable functions $\phi_0 : \{0,1\}^* \to \mathbb{G}_1^n$, $\phi_1 : \{0,1\}^* \to \mathbb{G}_2^n$ such that:*

$$\Pr\left[\begin{array}{c} \rho_0, \rho_1 \leftarrow \$, (k_0, k_1) \leftarrow\!\!\$\ \mathsf{PCG.Setup}(1^\lambda, \rho_0, \rho_1) \\ (R_0, S_0) \leftarrow \mathsf{PCG.Expand}(0, k_0), \\ (R_1, S_1) \leftarrow \mathsf{PCG.Expand}(1, k_1) \end{array} : \begin{array}{l} R_0 = \phi_0(\rho_0) \\ R_1 = \phi_1(\rho_1) \end{array} \right] \geq 1 - \mathsf{negl}(\lambda),$$

where $e : \mathbb{G}_1^n \times \mathbb{G}_2^n \to \mathbb{G}_T^n$ is the bilinear map applied componentwise.

- ***Programmable security.*** *The distributions:*

$$\left\{(\mathrm{k}_1, (\rho_0, \rho_1)) \mid \rho_0, \rho_1 \leftarrow \$, (\mathrm{k}_0, \mathrm{k}_1) \leftarrow\!\!\$\ \mathsf{PCG.Setup}(1^\lambda, \rho_0, \rho_1)\right\}$$

and

$$\left\{(\mathrm{k}_1, (\rho_0, \rho_1)) \mid \rho_0, \rho_1, \tilde{\rho}_0 \leftarrow \$, (\mathrm{k}_0, \mathrm{k}_1) \leftarrow\!\!\$\ \mathsf{PCG.Setup}(1^\lambda, \tilde{\rho}_0, \rho_1)\right\}$$

as well as:

$$\left\{(\mathrm{k}_0, (\rho_0, \rho_1)) \mid \rho_0, \rho_1 \leftarrow \$, (\mathrm{k}_0, \mathrm{k}_1) \leftarrow\!\!\$\ \mathsf{PCG.Setup}(1^\lambda, \rho_0, \rho_1)\right\}$$

and

$$\left\{(\mathrm{k}_0, (\rho_0, \rho_1)) \mid \rho_0, \rho_1, \tilde{\rho}_0 \leftarrow \$, (k_0, k_1) \leftarrow\!\!\$\ \mathsf{PCG.Setup}(1^\lambda, \tilde{\rho}_0, \rho_1)\right\}$$

are computationally indistinguishable.

3.4.7 Idealized Models in Cryptography

Idealized models are foundational tools in cryptographic analysis, enabling the study of cryptographic primitives under idealized assumptions. This section presents some among the most widely used models: *Random Oracle Model (ROM)*, the *Random Permutation Model (RPM)*, and the *Ideal Cipher Model*. These abstractions simplify security proofs by assuming random behaviors for hash functions and block ciphers.

3.4.7.1 Random Oracle Model (ROM) The *Random Oracle Model (ROM)*, introduced by Bellare and Rogaway [BR93], models hash functions as random oracles. In this abstraction, a hash function H is treated as a theoretical black box that, given an input x, returns a uniformly random output $H(x)$ of fixed length. Consistency is guaranteed, as repeated queries with the same input produce the same output, but queries with distinct inputs generate independent outputs.

The ROM is used in the analysis of cryptographic protocols that rely on hash functions. For example, the Fiat-Shamir transformation [FS87] leverages the ROM to convert interactive proofs into non-interactive ones, preserving the zero-knowledge property. Additionally, the ROM is the usual environment for analyzing the security of digital signature schemes [BR94]. However, it is crucial to note that the ROM is an abstraction: real-world hash functions, such as SHA-256 [Nat15a], are deterministic algorithms with predefined structures and then cannot fully emulate the behavior of a random oracle.

3.4.7.2 Random Permutation Model (RPM) The *Random Permutation Model (RPM)* models block ciphers as random permutations. In this framework, a block cipher $\mathcal{E}$ is assumed to behave as a randomly chosen bijection from the set of all possible permutations over a fixed-size domain. Unlike the ROM, which models hash functions as random mappings, the RPM ensures that $\mathcal{E}$ is a bijective function, ensuring that each input has a unique output and

vice versa: this bijectivity aligns with the requirements of encryption, where each plaintext should map to a unique ciphertext and be invertible with the correct key.

The RPM has been instrumental in the analysis of block cipher constructions, such as the Data Encryption Standard (DES) [Nat99] and the Advanced Encryption Standard (AES) [01]. While real-world block ciphers are designed to approximate as close as possible random permutations, they are constrained by structured designs and limited key sizes. However, the RPM provides a valuable abstraction for understanding their security against attacks like differential [BS91] and linear cryptanalysis [Mat94].

3.4.7.3 Ideal Cipher Model The *Ideal Cipher Model* extends the RPM by introducing key dependency. In this model, a block cipher $\mathcal{E}_k$ is treated as a family of random permutations indexed by the key k. Each key k defines an independent random permutation over the message space, and the encryption process corresponds to applying the permutation associated with k.

This model is particularly useful for analyzing cryptographic constructions that rely on keyed primitives, such as pseudorandom functions. For example, Feistel network constructions [LR88] and puncturable pseudorandom functions (PPRFs) leverage the Ideal Cipher Model to achieve provable security guarantees. The model has also been applied to multi-instance security and to analyze the robustness of schemes under related-key attacks.

Limitations of Idealized Models While idealized models such as the ROM, RPM, and Ideal Cipher Model provide a simplified framework for analyzing cryptographic constructions, their assumptions do not align perfectly with real-world implementations: real hash functions and block ciphers are deterministic and exhibit structural properties that deviate from ideal randomness.

3.5 Code-Based Cryptography

Code-based cryptography, first introduced by McEliece in 1978 [McE78], is one of the oldest approaches in the field of cryptographic design. Built upon the computational hardness of decoding linear codes, this paradigm represents a cornerstone of both classical and post-quantum cryptography. The central assumption underpinning code-based cryptography is the intractability of some coding theory problems, such as the Syndrome Decoding Problem, which has consistently resisted efficient solutions, even with quantum algorithms: this makes code-based cryptography a robust and practical candidate for post-quantum standardization.

3.5.1 Syndrome Decoding Assumption

The *Syndrome Decoding (SD) Assumption* is a foundational problem in coding theory and a critical assumption in code-based cryptography, particularly relevant for post-quantum cryptographic schemes. At a high level, given a weight parameter w, the syndrome decoding problem asks to find a solution of Hamming weight w (under the promise that it exists) to a random system of linear equations over $\mathbb{F}_2$. Formally, let Sparse_w^K denote the set of all vectors of Hamming weight w over $\mathbb{F}_2^K$. Then:

Definition 3.5.1 (Syndrome Decoding Problem)**.** *Let K, k, w be three integers, with $K > k > w$. The syndrome decoding problem with parameters (K, k, w) is defined as follows:*

- *(Problem generation) Sample $H \leftarrow\!\!\$\ \mathbb{F}_2^{k\times K}$ and $x \leftarrow\!\!\$\ \mathsf{Sparse}_w^K$. Set $y \leftarrow H \cdot x$. Output (H, y)*
- *(Goal) Given (H, y), find $x \in \mathsf{Sparse}_w^K$ such that $H \cdot x = y$.*

A pair (H, y) is referred to as an *instance* of the syndrome decoding problem. Variants of the syndrome decoding problem are also considered in this context, involving different restrictions on the solution vector x. The constraint on x can be rephrased as a linear equation over $\mathbb{N}$: the solution vector x must satisfy the condition $\langle x, \mathbf{1}\rangle = w$, where $\mathbf{1}$ denotes the all-1 vector, and the inner product is computed over the integers (this perspective is specifically applicable to syndrome decoding over $\mathbb{F}_2$). Other standard variants of syndrome decoding from the literature can similarly be interpreted as instances of a broader notion of *syndrome decoding under $\mathbb{N}$-linear constraints*, introduced below.

Definition 3.5.2 (Syndrome Decoding under $\mathbb{N}$-Linear Constraints)**.** *Let K, k, w, c be four integers, with $K > k > w$ and $k > c$. Let $L \in \mathbb{N}^{c\times K}$ be a matrix and $\mathbf{v} \in \mathbb{N}^c$ a vector; $(L, \mathbf{v})$ is referred to as the* $\mathbb{N}$-linear constraint*. The constraint $(L, \mathbf{v})$ is said* feasible *if it is possible to sample a uniformly random element from the set $\{x \in \{0,1\}^K \ : \ L \cdot x = \mathbf{v}\}$ in time $\mathsf{poly}(\lambda)(K)$.*

The syndrome decoding problem with parameters (K, k, w) and feasible $\mathbb{N}$-linear constraint $(L, \mathbf{v})$ is defined as follows:

- *(Problem generation) Sample a matrix $H \leftarrow\!\!\$\ \{0,1\}^{k\times K}$ and a vector $x \leftarrow\!\!\$\ \{x \in \{0,1\}^K \ : \ L \cdot x = \mathbf{v}\}$. Set $y \leftarrow H \cdot x \bmod 2$. Output (H, y).*
- *(Goal) Given (H, y), find $x \in \{0,1\}^K$ such that*
  - *$H \cdot x = y \bmod 2$ (the $\mathbb{F}_2$-linear constraint), and*
  - *$L \cdot x = \mathbf{v}$ over $\mathbb{N}$ (the $\mathbb{N}$-linear constraint).*

3.5.1.1 Regular Syndrome Decoding Problem - RSD

The regular syndrome decoding problem is a well-established variant of syndrome decoding: it was introduced in 2003 in [AFS03] as the assumption underlying the FSB candidate to the NIST hash function competition and was subsequently analyzed in [FGS07; MDC+11; BLP+11], among others. It has also been used and analyzed in many recent works on secure computation, such as [HOS+18; BCG+18; BCG+19b; BCG+19a; BCG+20b; YWL+20; WYK+21; RS21; CRR21; BCG+22]

In the RSD problem, the solution vector x is required not only to have a Hamming weight w, but also to exhibit a specific regular structure. Formally, a vector $x \in \mathbb{F}_2^K$ is said to be *w-regular* if it can be divided into w blocks, each containing exactly one non-zero position. This constraint adds an extra layer of complexity to the decoding process, as it limits the possible configurations of x even further.

Definition 3.5.3 (Regular Syndrome Decoding Problem)**.** *Let K, k, w be three integers, with $K > k > w$. The regular syndrome decoding problem with parameters (K, k, w) is defined as follows:*

- *(Problem generation) Sample $H \leftarrow\!\!\$\ \mathbb{F}_2^{k \times K}$ and $x \leftarrow\!\!\$\ \mathsf{Regular}_w^K$. Set $y \leftarrow H \cdot x$. Output (H, y)*
- *(Goal) Given (H, y), find $x \in \mathsf{Regular}_w^K$ such that $H \cdot x = y$.*

In particular, in the field of digital signatures, the RSD problem serves as the underlying hard problem in various code-based schemes that use the *Fiat-Shamir with Aborts* transformation and the *MPC-in-the-Head* paradigm, allowing for efficient zero-knowledge proofs without the need for a trapdoor. Examples include recent advancements in code-based signature schemes where the RSD assumption has been leveraged to achieve compact signatures with high levels of security.

CHAPTER 4

Signature via Fiat-Shamir

The chapter introduces the main cryptographic tools used in the results presented in this thesis. The focus begins with digital signatures and their natural extension to threshold schemes, emphasizing their relevance in post-quantum security [FJR22]. Key protocols and proof techniques, such as zero-knowledge proofs and the Fiat-Shamir heuristic [FS87], are discussed. A key role is played by the MPC-in-the-Head paradigm [IKO+07; KKW18], a fundamental tool enabling efficient post-quantum signatures through techniques like puncturable pseudorandom functions.

Contents

4.1 Digital Signatures

The concept of digital signatures was first introduced by Diffie and Hellman in their work [DH76], where they proposed the notion of public-key cryptography as a theoretical framework. Shortly thereafter, Rivest, Shamir, and Adleman formally instantiated digital signatures with the RSA scheme [RSA78], demonstrating their practical feasibility. These initial developments laid the foundation for a vast area of research, making digital signatures one of the most studied primitives in cryptography: they allow to ensure authenticity and integrity in electronic communications. Their security is based on the computational hardness of certain mathematical problems, such as integer factorization or discrete logarithms, which are exploited through one-way functions with trapdoors. The development of digital signature schemes has progressively advanced, addressing the need for improved security guarantees: modern signature schemes incorporate advanced cryptographic techniques, striking a balance between efficiency and robustness.

Since the NIST call for post-quantum cryptographic standards in 2016 [Natng], the focus has shifted to developing digital signatures secure against quantum adversaries. This has led to the instantiation and the analysis of numerous post-quantum schemes, built upon different security assumptions, such as the hardness of lattice problems, multivariate polynomial equations, hash-based constructions, and coding theory problems.

Definition 4.1.1 (Digital Signature Scheme)**.** *A digital signature scheme Σ is defined as a tuple of three probabilistic polynomial-time (PPT) algorithms* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*, specified as follows:*

- $\mathsf{KeyGen}(1^\lambda)$*: A randomized key generation algorithm that takes as input the security parameter λ and outputs a signing key* sk *and a verification key* pk*. Formally,* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*.*
- $\mathsf{Sign}(\mathsf{sk}, m)$*: A signing algorithm that, given the private signing key* sk *and a message $m \in \mathcal{M}$, outputs a signature σ. Formally, $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$.*
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$*: A deterministic verification algorithm that, given the public verification key* pk*, a message $m \in \mathcal{M}$, and a signature σ, outputs a bit $b \in \{0, 1\}$, where $b = 1$ indicates that the signature is valid. Formally, $b \leftarrow \mathsf{Verify}(\mathsf{pk}, m, \sigma)$.*

A digital signature scheme must satisfy the following security properties:

Definition 4.1.2 (Correctness)**.** *A digital signature scheme Σ is correct if, for any* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and any message $m \in \mathcal{M}$, it holds that:*

$$\mathsf{Verify}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1.$$

Standard security notions for signature schemes are existential unforgeability against key-only attacks (EUFKO, Definition 4.1.3) and against chosen-message attacks (EUF-CMA, Definition 4.1.4).

Definition 4.1.3 (EUF-KO security)**.** *Given a signature scheme* $\mathsf{Sig} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ *and security parameter* λ*,* Sig *is said to be* EUFKO*-secure if any PPT algorithm* $\mathcal{A}$ *has negligible advantage in the EUF-KO game, defined as*

$$\mathsf{Adv}^{\mathsf{EUFKO}}_{\mathcal{A}} = \Pr\left[\begin{array}{c} \mathsf{Verify}(\mathsf{pk}, \mu^*, \sigma^*) = 1 \end{array} \quad \begin{array}{c} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(\{0,1\}^\lambda) \\ (\mu^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{pk}) \end{array}\right].$$

Definition 4.1.4 (EUF-CMA security)**.** *Given a signature scheme* $\mathsf{Sig} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ *and security parameter* λ*,* Sig *is said to be* EUF-CMA*-secure if any PPT algorithm* $\mathcal{A}$ *has negligible advantage in the* EUF-CMA *game, defined as*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}} = \Pr\left[\begin{array}{c} \mathsf{Verify}(\mathsf{pk}, \mu^*, \sigma^*) = 1 \\ \wedge \mu^* \notin Q \end{array} \quad \begin{array}{c} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(\{0,1\}^\lambda) \\ (\mu^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk}) \end{array}\right],$$

where $\mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}$ *denotes* $\mathcal{A}$*'s access to a signing oracle with private key* sk *and* Q *denotes the set of messages* μ *that were queried to* $\mathsf{Sign}(\mathsf{sk}, \cdot)$ *by* $\mathcal{A}$*.*

4.1.1 Applications

Digital signatures find extensive application across various domains, ensuring authenticity and integrity in numerous practical systems [TSZ19]:

- **Financial Systems:** digital signatures are crucial in securing financial transactions, enabling verification of the sender's identity and preventing unauthorized modifications.
- **Critical Infrastructure:** in sectors such as energy and public transportation, digital signatures confirm the authorization of commands issued by the central control system.
- **Organizational Use:** enterprises and government agencies employ digital signatures to guarantee the integrity and confidentiality of sensitive documents, ensuring that they remain unchanged during storage and transmission.

4.1.2 Digital Signatures in Post-Quantum Cryptography

The advent of post-quantum cryptography, accompanied by the quest for cryptographic primitives resilient to quantum attacks, has naturally extended to digital signature schemes

[TSZ19]. The primary challenge in designing post-quantum digital signatures lies in balancing security and efficiency: these schemes must be constructed based on assumptions that remain secure against adversaries with quantum computational capabilities, while simultaneously maintaining high performance to align with the practical demands of their numerous applications.

The call for the National Institute of Standards and Technology (NIST) Post-Quantum Cryptography Standardization Project [Natng] has significantly accelerated the standardization of such schemes. Numerous protocols have been proposed , relying on a variety of mathematical assumptions, including lattice-based problems, multivariate polynomial systems, hash-based constructions, and code-based assumptions. Each approach presents distinct trade-offs concerning key size, signature size, and computational complexity.

4.2 Threshold Signatures

Threshold signatures extend the classical concept of digital signatures to a distributed multi-party setting. Introduced to address the limitations of single-signer schemes, these schemes enable a subset of t or more parties, out of a total n, to jointly generate a single valid signature. This generalization was first explored in the context of threshold Schnorr signatures [GJK+07; AF04; SS01], which demonstrated robustness even in the presence of misbehaving parties.

Historical Overview Early threshold signature schemes relied heavily on distributed key generation protocols [CGJ+99; AF04] and secure erasures to achieve adaptive security. Notably, Abe-Fehr [AF04] introduced adaptively secure threshold Schnorr signatures without relying on erasures but required the threshold t to be strictly less than $n/2$. Subsequent works improved upon this by addressing inefficiencies and exploring asynchronous settings [GJK+07; CGJ+99; KRT24; CKM23; BCK+22].

In recent years, significant advancements have been made in constructing adaptively secure threshold signatures without relying on impractical assumptions: techniques such as the single inconsistent player (SIP) [JL00] and other adaptively secure models [CKM23; BLS+24] have played a key role in achieving strong security guarantees under standard assumptions. Recent threshold schemes [PL22; DR24] rely on assumptions like the AGM and OMDL, ensuring tight security proofs.

Threshold Schnorr signatures continue to gain attention due to their simplicity and efficiency [CKM23]. They provide a foundation for many modern threshold protocols, including those relying on lattice-based constructions [EKT24; GKS24], and multi-signature variants. This progress highlights the increasing importance of tightly secure threshold signatures in applications such as blockchain systems, distributed key management, and secure multi-party computation.

The notion of tight security proofs has been a critical area of research for threshold signatures [PL22; KRT24]. Threshold ECDSA signatures, for example, have been formally analyzed under strict conditions for $t = n - 1$ [CGG+20]. In parallel, threshold schemes have evolved to accommodate adversarial models where signers can be adaptively corrupted, requiring resilience against partial forgeries and strong unforgeability guarantees [BLS+24].

Definition 4.2.1 (Threshold Signature Scheme)**.** *A* threshold signature scheme TS *is defined as a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Combine}, \mathsf{Verify})$ *satisfying the following:*

- $\mathsf{Setup}(1^\lambda)$*: Generates the public system parameters* params*, where* λ *is the security parameter.*
- $\mathsf{KeyGen}(\mathsf{params}) \to (\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^n)$*: Generates a public key* pk *and a set of secret key shares* $\{\mathsf{sk}_i\}_{i=1}^n$ *for* n *participants.*
- $\mathsf{Sign}(\mathsf{sk}_i, m) \to \sigma_i$*: Given a secret key share* sk_i *and a message* m*, each participant* P_i *outputs a partial signature* σ_i*.*
- $\mathsf{Combine}(\{\sigma_i\}_{i\in S}, \mathsf{params}) \to \sigma$*: A deterministic algorithm that combines partial signatures* $\{\sigma_i\}_{i\in S}$ *from a qualified set* $S \subseteq [n]$ *with* $|S| \geq t$ *into a final signature* σ*.*
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma) \to \{0, 1\}$*: Given a public key* pk*, a message* m*, and a signature* σ*, outputs 1 if* σ *is valid for* m *under* pk*, and 0 otherwise.*

A threshold signature scheme must satisfy the following property: for any $(S, \mathsf{pk}, \{\mathsf{sk}_i\}_{i\in S}) \leftarrow \mathsf{KeyGen}(\mathsf{params})$ *and any message* m*, it holds that:*

$$\Pr\left[\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1 \mid \sigma \leftarrow \mathsf{Combine}(\{\mathsf{Sign}(\mathsf{sk}_i, m)\}_{i\in S}, \mathsf{params})\right] = 1.$$

4.2.1 Security Model

The security of a threshold signature scheme is formalized in the presence of adversarial participants through the concept of *existential unforgeability under chosen message attacks* (TS-EUF-CMA). The security game models a scenario where an adversary can adaptively corrupt signers, query signing oracles, and attempt to forge signatures [BLT+24].

Definition 4.2.2 (TS-EUF-CMA Security)**.** *Let* $\mathsf{TS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Combine}, \mathsf{Verify})$ *be a* (t, n)*-threshold signature scheme.* TS *is existentially unforgeable under chosen message attacks if, for all PPT adversaries* $\mathcal{A}$*, it holds that*

$$\mathrm{Adv}^{\text{TS-EUF-CMA}}_{\mathcal{A},\mathsf{TS}}(\lambda) = \Pr\left[\text{TS-EUF-CMA}_{\mathcal{A},\mathsf{TS}}(\lambda) = 1\right] \leq \epsilon$$

where ϵ *is negligible in the security parameter* λ*.*

The game TS-EUF-CMA$_{\mathcal{A},\mathsf{TS}}$ is defined in Protocol 1

Game TS-EUF-CMA$_{\mathcal{A},\mathbf{TS}}(\lambda)$

1: $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$
2: $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{Gen}(\mathsf{par})$
3: $\mathsf{Sign} := (\mathsf{Next}, \mathsf{Sign}_0, \mathsf{Sign}_1, \mathsf{Sign}_2)$
4: $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign},\mathrm{Corrupt}}(\mathsf{par}, \mathsf{pk})$
5: **if** $m^* \in$ Queried **then**
6: **return** 0
7: **end if**
8: **return** $\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*)$

*Protocol 1: The game **TS-EUF-CMA** for a (t, n)-threshold signature scheme* TS.

Oracles

Oracle Corrupt(i)
1: **if** $|\text{Corrupted}| \geq t$ **then**
2: **return** $\perp$
3: **end if**
4: Corrupted := Corrupted $\cup \{i\}$
5: **return** $(\mathsf{sk}_i, \text{state}[\cdot, i])$

Oracle Next(sid, S, m)
1: **if** $|S| \neq t + 1 \vee S \not\subseteq [n]$ **then**
2: **return** $\perp$
3: **end if**
4: **if** $sid \in$ Sessions **then**
5: **return** $\perp$
6: **end if**
7: Sessions := Sessions $\cup \{sid\}$
8: message$[sid]$:= m, signers$[sid]$:= S
9: Queried := Queried $\cup \{m\}$
10: **for** $i \in S$ **do**
11: round$[sid, i]$:= 0
12: **end for**
13: **return** state$[\cdot, i]$

Oracle $\mathsf{Sign}_0(sid, i)$
1: **if** $\text{Allowed}(sid, i, 0, \perp) = 0$ **then**
2: **return** $\perp$
3: **end if**
4: S := signers$[sid]$, m := message$[sid]$

5: $(\text{pm}, \text{St}) \leftarrow \mathsf{Sign}_0(S, i, \mathsf{sk}_i, m)$
6: $\text{pm}_1[sid, i] := \text{pm}, \text{state}[sid, i] := \text{St}$
7: $\text{round}[sid, i] := 1$
8: **return** pm

Oracle $\mathsf{Sign}_1(sid, i, \mathcal{M}_1)$

1: **if** $\text{Allowed}(sid, i, 1, \mathcal{M}_1) = 0$ **then**
2: **return** $\perp$
3: **end if**
4: $(\text{pm}, \text{St}) \leftarrow \mathsf{Sign}_1(\text{state}[sid, i], \mathcal{M}_1)$
5: $\text{pm}_2[sid, i] := \text{pm}, \text{state}[sid, i] := \text{St}$
6: $\text{round}[sid, i] := 2$
7: **return** pm

Oracle $\mathsf{Sign}_2(sid, i, \mathcal{M}_2)$

1: **if** $\text{Allowed}(sid, i, 2, \mathcal{M}_2) = 0$ **then**
2: **return** $\perp$
3: **end if**
4: $\text{pm} \leftarrow \mathsf{Sign}_2(\text{state}[sid, i], \mathcal{M}_2)$
5: $\text{round}[sid, i] := 3$
6: **return** pm

Algorithm Allowed$(sid, i, r, \mathcal{M})$

1: **if** $sid \notin \text{Sessions}$ **then**
2: **return** 0
3: **end if**
4: $S := \text{signers}[sid], H := S \setminus \text{Corrupted}$
5: **if** $i \notin H$ **then**
6: **return** 0
7: **end if**
8: **if** $\text{round}[sid, i] \neq r$ **then**
9: **return** 0
10: **end if**
11: **if** $r > 0$ **then**
12: **parse** $(\text{pm}_i)_{i \in S} := \mathcal{M}$
13: **if** $\text{pm}_i \neq \text{pm}_r[sid, i]$ **then**
14: **return** 0
15: **end if**
16: **end if**
17: **return** 1

Protocol 2: All the oracles needed in Game 1

The game models realistic adversarial behavior:

- **Adaptive Corruption:** $\mathcal{A}$ may corrupt up to $t-1$ signers, learning their secret key shares and states. This models adversaries that adaptively compromise system participants.
- **Signing Queries:** $\mathcal{A}$ can request partial signatures for chosen messages from any uncorrupted signer, simulating legitimate signing interactions.
- **Forgery Attempt:** $\mathcal{A}$ outputs a candidate forgery (m^*, σ^*), aiming to produce a valid signature on a message m^* without access to sufficient key shares.

The scheme remains secure if $\mathcal{A}$ cannot generate a valid signature outside the constraints of the game.

4.2.2 Applications

Threshold signatures have numerous applications since they can be used in decentralized systems –for example multi-signature transactions–. Recent developments in adaptive security and tight proofs [BCK+22; CKM23] have significantly enhanced their applicability in modern cryptographic systems. Below, three fundamental use cases where threshold signatures provide significant benefits are outlined:

- **Blockchain and Multi-Signature Transactions:** threshold signatures enable secure multi-party authorization without relying on a single trusted signer [Nak08; Woo14]. This allows decentralized custody and robust security for assets in blockchain systems.
- **Secure Voting Protocols:** in electronic voting systems, threshold signatures ensure collective authorization of election outcomes while preserving anonymity.
- **Distributed Key Management:** threshold signatures provide a solution for protecting sensitive keys across multiple entities [Sho00]. Certificate Authorities (CAs) and other security-critical systems use threshold techniques to distribute trust and minimize single points of failure.

4.3 Protocols and Proof Techniques

4.3.1 Interactive proofs

Interactive proofs, introduced and analyzed in two foundational works by Babai [Bab85] and Goldwasser, Micali, and Rackoff [GMR85], provide a framework where a prover and a verifier interact to establish the validity of a statement, generalizing the verification process used in NP [1] proofs. These systems generalize traditional proof verification by allowing the verifier to interact with the prover adaptively, using randomness, and ensuring correctness.

[1] The complexity class NP includes languages for which a computationally unbounded prover can provide a proof of membership that a polynomial-time verifier can efficiently check.

Definition 4.3.1. *A language $\mathcal{L}$ is in the class* NP *if there exists a polynomial-time algorithm $R_{\mathcal{L}}$ such that:*

$$\mathcal{L} = \{x \mid \exists \pi, |\pi| = \mathsf{poly}(|x|) \wedge R_{\mathcal{L}}(x, \pi) = 1\}.$$

The string π is called the *witness* for the statement of $x \in \mathcal{L}$. This captures the essence of NP: the ability to verify efficiently when the correct witness is given.
Interactive proofs enhance this model by incorporating two key features:

- Both the prover and the verifier can use randomness, and the verifier's decision must be correct with high probability.
- The prover does not present a fixed witness but interacts with the verifier, responding to queries in an exchange.

Definition 4.3.2. *An n-round interactive proof system $(\mathcal{P}, \mathcal{V})$ for a language $\mathcal{L}$ consists of two randomized algorithms: a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, both modeled as probabilistic polynomial-time Turing machines. The verifier $\mathcal{V}$ interacts with $\mathcal{P}$ and either accepts or rejects the input x. It holds that:*

- ***Completeness.*** *The system is complete if, for any $x \in \mathcal{L}$:*

$$\Pr[(\mathcal{P}, \mathcal{V})(x) = 1] \geq 1 - \epsilon.$$

- ***Soundness.*** *The system is sound if, for any $x \notin \mathcal{L}$ and for any (possibly malicious) prover $\mathcal{P}'$:*

$$\Pr[(\mathcal{P}', \mathcal{V})(x) = 1] \leq \epsilon.$$

In this case, the quantity ϵ is called soundness error of the interactive proof.

Hence, a smaller soundness error indicates a higher level of security. For instance, in practical applications, proof systems are often designed to achieve negligible soundness error (e.g., $\epsilon = 2^{-\lambda}$, where λ is a security parameter) by increasing the number of repetitions. Soundness error is a key metric in the design of cryptographic protocols, ensuring that the system resists forgery and deception by untrustworthy provers.

4.3.2 Zero-Knowledge Interactive Proofs

Zero-knowledge interactive proofs extend interactive proofs by ensuring that no information, apart from the validity of the statement, is leaked to the verifier.

Definition 4.3.3. *An interactive proof system $(\mathcal{P}, \mathcal{V})$ for a language $\mathcal{L}$ is zero-knowledge if there exists a simulator* Sim *such that the transcript of the interaction is computationally indistinguishable from the output of* Sim. *An interactive zero-knowledge proof satisfies:*

- ***Completeness.*** *For $x \in \mathcal{L}$:*

$$\Pr[(\mathcal{P}, \mathcal{V})(x) = 1] \geq 1 - \epsilon.$$

- ***Soundness.*** *For $x \notin \mathcal{L}$ and any $\mathcal{P}'$:*

$$\Pr[(\mathcal{P}', \mathcal{V})(x) = 1] \leq \epsilon.$$

- ***Zero-Knowledge.*** *For every PPT verifier $\mathcal{V}^*$, there exists a simulator* Sim *such that:*

$$\mathsf{View}_{\mathcal{V}^*}[(\mathcal{P}, \mathcal{V}^*)(x)] \approx \mathsf{Sim}(x).$$

Interactive zero-knowledge proofs extend interactive proofs by adding the zero-knowledge property, ensuring that no information beyond the validity of the statement is revealed to the verifier. This property can be classified based on the indistinguishability notion used. Zero-knowledge proofs can be categorized as follows:

- **Perfect Zero-Knowledge.** The transcript of the interaction is *identically distributed* to the output of a simulator Sim. This provides the strongest form of zero-knowledge.
- **Statistical Zero-Knowledge.** The transcript and the simulator's output are statistically close, meaning their distance is negligible but non-zero. This type is weaker than perfect zero-knowledge.
- **Computational Zero-Knowledge.** The transcript and the simulator's output are computationally indistinguishable under the assumption that the verifier is computationally bounded (PPT). This is the weakest but most commonly used form in cryptographic applications.

4.3.2.1 Honest-Verifier Zero-Knowledge Proof

An interactive proof is honest-verifier zero-knowledge (HVZK) if the zero-knowledge property holds against an honest verifier who follows the protocol correctly: for every honest verifier $\mathcal{V}$, there exists a simulator Sim such that the transcript is indistinguishable from the output of Sim.

4.3.3 Proofs of Knowledge

Proofs of knowledge formalize the notion of demonstrating not only the existence of a witness w for a statement $x \in \mathcal{L}$ but also ensuring that the prover "knows" this witness. These concepts are central to cryptographic protocols that require assurance of knowledge rather than just mere existence.
To define this concept rigorously, it's necessary to clarify what it means for a prover to "know" a witness. Intuitively, a machine is said *to know* a value w if there exists an efficient algorithm, called an *extractor*, that can compute w when given access to the prover.

Definition 4.3.4. *A party $\mathcal{P}^*$ knows a witness w for a statement $x \in \mathcal{L}$ if there exists an efficient algorithm* Ext*, given access to $\mathcal{P}^*$, that outputs w such that $R_{\mathcal{L}}(x, w) = 1$. The extractor's runtime should be inversely related to the success probability of $\mathcal{P}^*$.*

This property transforms the traditional soundness guarantee into a *knowledge-extraction* property, ensuring that any prover convincing a verifier can be used to efficiently recover a valid witness.

Definition 4.3.5. *For every efficient prover $\mathcal{P}^*$ such that $\Pr[(\mathcal{P}^*, \mathcal{V})(x) = 1] \geq \epsilon$, there exists an efficient extractor* Ext *that outputs w satisfying $R_{\mathcal{L}}(x, w) = 1$.*

4.3.4 Zero-Knowledge Proofs of Knowledge

Zero-knowledge proofs of knowledge extend the concept of proofs of knowledge by incorporating the zero-knowledge property, ensuring that no information about the witness is revealed during the proof process. Informally, a zero-knowledge proof of knowledge is an interactive protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, where the verifier is convinced that the prover knows a witness w for a given statement $x \in \mathcal{L}$ while learning nothing beyond it. For a formal definition, given a two-party interactive protocol between PPT algorithms, A with input a and B with input b where only B gets an output, two random variables are introduced: $\langle A(a), B(b)\rangle$ denotes the output of the protocol, and $\textsc{View}(A(a), B(b))$ denotes the transcript of the protocol.

Definition 4.3.6. *A honest-verifier zero-knowledge argument of knowledge with soundness error ε for a NP language*

$$\mathcal{L} = \{x \in \{0,1\}^* \ : \ \exists w, (x, w) \in \mathcal{R}_{\mathcal{L}} \wedge |w| = \mathsf{poly}(\lambda)(|x|)\}$$

with relation $\mathcal{R}_{\mathcal{L}}$ is a n-ronud interactive protocol between a prover P and a verifier V which satisfies the following properties:

- ***Perfect Completeness:*** *for every $(x, w) \in \mathcal{R}_{\mathcal{L}}$, the verifier always accepts the interaction with an honest prover:*
$$\Pr[\langle \mathsf{P}(x, w), \mathsf{V}(x)\rangle = 1] = 1.$$
- ***ε-Soundness:*** *[BG93] for every PPT algorithm $\tilde{\mathsf{P}}$ such that*
$$\Pr[\langle \tilde{\mathsf{P}}(x), \mathsf{V}(x)\rangle = 1] = \tilde{\varepsilon} > \varepsilon,$$
there exists an extractor *algorithm $\mathcal{E}$ which, given rewindable black-box access to $\tilde{\mathsf{P}}$, outputs a w' satisfying $\mathcal{R}_{\mathcal{L}}(x, w') = 1$ in time $\mathsf{poly}(\lambda)(\lambda, (\tilde{\varepsilon} - \varepsilon)^{-1})$ with probability at least $1/2$.*

- ***Zero-Knowledge:*** *an argument of knowledge is (computationally, statistically, perfectly) zero knowledge if there exists a PPT simulator* Sim *such that for every* $(x, w) \in \mathcal{R}_{\mathcal{L}}$*,*

$$\mathsf{Sim}(x) \equiv \mathit{View}(\mathsf{P}(x, w), \mathsf{V}(x)),$$

where $\equiv$ *denotes computational, statistical, or perfect indistinguishability between the distributions.*

The knowledge error λ reflects the deviation from perfect knowledge extraction and can be reduced to negligible levels through repetition.

4.3.4.1 Honest Verifier Zero-Knowledge Proof of Knowledge

Gap-HVZK A *gap* honest-verifier zero-knowledge argument of knowledge [CKY09] with gap $\mathcal{L}'$, where $\mathcal{L}' \supseteq \mathcal{L}$ is an NP language with relation $\mathcal{R}_{\mathcal{L}'}$, is defined as an honest-verifier zero-knowledge argument of knowledge, with the following relaxation of ε-soundness: the extractor $\mathcal{E}$ is only guaranteed to output a witness w' such that $(x, w') \in \mathcal{L}'$. Concretely, in the studied setting, the witness is a valid solution to the syndrome decoding problem, and the language $\mathcal{L}'$ contains all strings which are *sufficiently close* (in a well-defined sense) to a valid solution. This is similar in spirit to the notion of *soundness slack* often used in the context of lattice-based zero-knowledge proof, where the honest witness is a vector with small entries, and the extracted vector can have significantly larger entries.

Honest-but-Curious Adversary. An honest-but-curious adversary, also known as a semi-honest adversary, is an entity in a cryptographic protocol that follows the protocol's instructions correctly but attempts to learn additional information from the messages it receives. While such an adversary does not deviate from the prescribed protocol, it may analyze data passed through the protocol to infer private information. Security against honest-but-curious adversaries is often achieved by ensuring that intermediate values are computationally indistinguishable from random, preventing the adversary from gaining unauthorized insights. This model is used in multi-party computation (MPC) and secure computation settings where participants are assumed to be passive but inquisitive.

Malicious Adversary. A malicious adversary is an adversary that may deviate arbitrarily from the protocol's instructions, including sending incorrect or malformed messages, modifying computations, or working with other adversaries to compromise the protocol's security. Protection against malicious adversaries requires more robust security measures to ensure that incorrect behavior can be detected and mitigated. Hence protocols secure against malicious adversaries are typically more complex and resource-intensive than those secure against honest-but-curious adversaries, as they must account for any possible deviation or manipulation by participants: they use more complex strategies including zero-knowledge proofs, consistency checks, or verifiable computation.

4.3.5 Fiat-Shamir Heuristic

The Fiat-Shamir heuristic [FS87] provides a method to transform interactive zero-knowledge proof systems 4.3.2 Σ into non-interactive zero-knowledge proofs (NIZKs). Specifically, it allows a prover $\mathcal{P}$ to prove membership of a word x in a language $\mathcal{L}$ by eliminating the need for interaction: for this reason, it has been adapted to numerous ZKPoK (see Section 4.3.4), obtaining zero-knowledge-based signatures. The Fiat-Shamir paradigm is typically applied to interactive proof systems with multiple rounds and in the simplest case -*i.e.* a three rounds protocol- the process proceeds at a high level in this way: the prover first computes the initial flow of the protocol Σ, denoted as c, which corresponds to the commitments. Then, a challenge e is computed as $e \leftarrow H(x, c)$, where H represents a cryptographic hash function. The prover completes the protocol using e as the challenge, generating the final response. While the Fiat-Shamir heuristic is widely used due to its efficiency, its security cannot be guaranteed under any standard assumption for hash functions. Instead, the heuristic can be proven secure in the *random oracle model* (see Section 3.4.7.1), where H is modeled as an idealized random function. However, this introduces a gap between theory and practice, as random oracles cannot be instantiated in practice due to their exponential size.

4.3.5.1 Transition from 5-Round ZKPoK to Signatures In this manuscript, this paradigm will be extensively used in its slightly more complex form, which allows for working with 5-round protocols.
For a NP language $\mathcal{L} = \{x \in \{0,1\}^* \ : \ \exists w, (x, w) \in \mathcal{R}_{\mathcal{L}} \wedge |w| = \mathsf{poly}(\lambda)(|x|)\}$ with instance x and witness w, the transformation of a 5-round zero-knowledge protocol (P, V) into a signature scheme can be outlined as follows:

Fiat-Shamir Transform on a 5-Round Protocol

$\mathsf{KeyGen}(1^\lambda)$: **Inputs:** A security parameter λ.

Generate a key pair $(\mathsf{pk}, \mathsf{sk})$ based on the structure of the underlying ZK protocol.

Outputs: The public key pk and private key sk.

$\mathsf{Sign}(\mathsf{sk}, m)$: **Inputs:** A secret key sk and a message m.

1. Simulate the prover's role in the (P, V) protocol:
   (a) Generate a first commitment c_1 using the witness w and a randomness r_1.
   (b) Compute the first challenge as $ch_1 \leftarrow H(c_1, m)$ using a cryptographic hash function H.
   (c) Generate a second commitment c_2 based on ch_1 and an additional randomness r_2.
   (d) Compute the second challenge as $ch_2 \leftarrow H(c_1, c_2, m)$.
   (e) Compute the final response R based on w, c_1, c_2, ch_1, ch_2, and all the used randomness.

2. Obtain the signature depending on how it is defined in the underlying ZKpoK $\sigma \leftarrow (c_1, c_2, R)$.

Outputs: The signature σ on the message m.

Verify(pk, m, σ): **Inputs:** A public key pk, a message m, and a signature $\sigma = (c_1, c_2, R)$.

1. Recompute the challenges:
   (a) Compute $ch_1 \leftarrow H(c_1, m)$.
   (b) Compute $ch_2 \leftarrow H(c_1, c_2, m)$.
2. Verify R by checking the consistency of commitments and responses according to the verification rules of the original 5-round protocol.

Outputs: 1 if the verification succeeds, 0 otherwise.

Signature schemes based on the Fiat-Shamir paradigm exhibit performance characteristics inherited from the underlying ZK protocol.
To obtain low soundness error for the security of the signature defined by the Fiat-Shamir heuristic, multiple parallel repetitions of the base ZKPoK are required. For example, traditional ZKPoKs, like Stern's protocol, have a per-round soundness error of $\frac{2}{3}$: this implies that to achieve a negligible error of 2^{-128}, the protocol must be repeated τ times, where: $\tau \geq \log_{2/3}(2^{-128}) \approx 216$. Of course, this repetition creates substantial communication and computational overhead.
The use of an MPC-in-the-Head (MPCitH) protocol in the definition of the ZKPoK addresses this problem by simulating a secure multiparty computation (MPC) within the prover's head, where virtual parties jointly emulate the protocol. The prover then commits to the parties' views, and the verifier requests partial openings to verify correctness. This approach compresses the communication cost by reducing the data that needs to be transmitted and verified while maintaining the desired security guarantees.
Furthermore, as it will be argued in Chapter 3, using tools like puncturable pseudorandom functions (PPRFs) and GGM trees allows MPCitH to represent and verify iterations so efficiently that $\tau = 8$ to $\tau = 16$ repetitions are sufficient to achieve 128-bit security, drastically reducing the overhead compared to traditional methods.

4.4 MPC and MPC-in-the-Head

The MPC-in-the-head paradigm, introduced in the seminal work of [IKO+07], provides a compiler that, given an n-party secure computation protocol for computing a function f' in the honest-but-curious model, produces an honest-verifier zero-knowledge argument of knowledge of x such that $f(x) = y$, for some public y, where f' is a function related to f.
At a high level (and specializing to MPC in the head with all-but-one additive secret sharing – the original compiler is more general), the compiler proceeds by letting the prover additively share the witness x into $(x_1, \cdots, x_n)$ among n virtual parties $(P_1, \cdots, P_n)$, run in his head

an MPC protocol for securely computing $f'(x_1, \cdots, x_n) = f(\sum_i x_i)$ (where the sum is over some appropriate ring), and commit to the views of all parties. Then, the verifier queries a random size-$(n-1)$ subset of all views, which the prover opens. The verifier checks that these views are consistent and that the output is correct She accepts if all checks succeed. Soundness follows from the fact that the MPC protocol is correct, hence if the prover does not know a valid x, one of the views must be inconsistent with the output being correct (the soundness error is therefore $1/n$). Honest-verifier zero-knowledge follows from the fact that the MPC protocol is secure against passive corruption of $n-1$ parties (and the fact that $n-1$ shares of x leak no information about x).

Chapter 5

Post-Quantum Signatures from RSD

This chapter introduces a post-quantum digital signature scheme based on the Regular Syndrome Decoding assumption [AFS03; FGS07]. A 5-round zero-knowledge proof system using the MPC-in-the-Head paradigm [IKO+07; FJR22] is described and compiled into a compact signature via the Fiat-Shamir heuristic [FS87]. A key contribution is the analysis of soundness in a relaxed setting, allowing almost regular witnesses, together with an exploration of the RSD assumption and related cryptanalytic attacks [CRR21; BCG+22]. The resulting scheme achieves competitive performance in size and efficiency.

Contents

Notation. Given an integer $n \in \mathbb{N}$, the set $\{1, \cdots, n\}$ will be denoted by $[n]$. Bold lowercase will be used for vectors and uppercase for matrices. Given a vector $\mathbf{v} \in \mathbb{F}^n$ and a permutation $\pi : [n] \mapsto [n]$, $\pi(\mathbf{v})$ is used to denote the vector $(v_{\pi(1)}, v_{\pi(2)}, \cdots, v_{\pi(n)})$. For $\mathbf{u}, \mathbf{v} \in \{0,1\}^n$, $\mathbf{u} \oplus \mathbf{v}$ represents the bitwise-XOR of $\mathbf{u}$ and $\mathbf{v}$, $\mathbf{u} \odot \mathbf{v}$ represents the bitwise-AND (also called Schur product, or Hadamard product) of $\mathbf{u}$ and $\mathbf{v}$, and $\mathsf{HW}(\mathbf{u})$ denotes the Hamming weight of $\mathbf{u}$ (*i.e.* its number of nonzero entries). For a set S, $s \leftarrow\!\$\ S$ indicates that s is sampled uniformly from S. For a probabilistic Turing machine $\mathcal{A}$ and an input x, $y \leftarrow\!\$\ \mathcal{A}(x)$ indicates that y is sampled by running $\mathcal{A}$ on x with a uniform random tape, or $y \leftarrow \mathcal{A}(x; r)$ when the random coin is made explicit.
Given a vector $\mathbf{u} \in \mathbb{Z}_T^\ell$ and an integer T, writing $(\mathbf{u_1}, \cdots, \mathbf{u_n}) \leftarrow\!\$\ [\![\mathbf{u}]\!]_T$ indicate that the vectors $\mathbf{u_i}$ (called the *i-th additive share* of $\mathbf{u}$) are sampled uniformly at random over $\mathbb{Z}_T^\ell$ conditioned on $\sum_i \mathbf{u_i} = \mathbf{u}$. This notation is sometimes abused by writing $[\![\mathbf{u}]\!]_T$ to denote the tuple $(\mathbf{u_1}, \cdots, \mathbf{u_n})$. For a vector $\mathbf{v} \in \{0,1\}^\ell$, $[\![\mathbf{v}]\!]_T$ with $T > 2$ will be written using the natural embedding of $\{0,1\}$ into $\mathbb{Z}_T$.

5.1 Problem Statement and Model

While it will be used the MPC in the head paradigm, as in previous works [GPS21; FJR21; BGK+22; FJR22], the choice of the underlying MPC protocol departs significantly from all previous work: the starting point is the observation that checking $H \cdot x = y$ and checking the structure of x can each be done using linear operations, over $\mathbb{F}_2$ for the former, and over $\mathbb{Z}$ for the latter. In standard MPC protocol, linear operations over a ring $\mathcal{R}$ are usually "for free", provided that the values are shared over $\mathcal{R}$. Therefore, the only component that requires communication is a *share conversion* mechanism, to transform shares over $\mathbb{F}_2$ into shares over a larger integer ring. A share conversion protocol is introduced, which exhibits very good performance. However, the presented protocol works in the preprocessing model, where the parties are initially given a correlated random string by a trusted dealer. The use of preprocessing in the MPC in the head paradigm has appeared in previous works [KKW18; GPS21], and handling the preprocessing phase usually incurs a significant communication overhead (due to the need to check that the prover correctly emulated it).
Nevertheless, a core technical contribution of this chapter is a method, tailored to the chosen setting, to handle the preprocessing phase *for free* (*i.e.* without incurring any communication overhead). At a high level, it is possible to achieve this by letting the verifier *randomly shuffle* the preprocessing strings, instead of verifying them. A careful and non-trivial combinatorial analysis shows that a cheating prover has a very low probability of providing an accepting proof for *any* choice of the initial (pre-permutation) preprocessing strings, over the choice of the verifier permutation. Furthermore, it is possible to observe that the cheating probability becomes much lower by focusing on cheating provers using a witness that is *far* from a regular witness (in the sense that it has multiple non-weight-1 blocks). For an appropriate setting of the parameters, the hardness of finding solutions *close* to regular witnesses becomes equivalent to the standard regular syndrome decoding assumption (where the solution must be strictly regular), hence this relaxation of the soundness still yields a signature scheme (after compilation with Fiat-Shamir) whose security reduces to the standard RSD assumption.

To complement the analysis, an analysis of the RSD assumption is also provided with a focus on the relation between RSD and the standard syndrome decoding assumption depending on the parameter regime, reviewing existing attacks on RSD from the literature, fine-tuning, and improving the attacks on several occasions. Eventually, a new "adversary-optimistic" attack against RSD is also developed, showing how a linear-time solution to the approximate birthday problem would yield faster algorithms for RSD (in the parameter choices, it is assumed that such an algorithm exists for the sake of choosing conservative parameters).

5.1.1 Performances

While analyzing the proposed approach is relatively involved, the protocol structure is extremely simple. The computation of the zero-knowledge proof is mostly dominated by simple XORs, calls to a length-doubling PRG (which can be instantiated very efficiently from AES over platforms with hardware support for AES), and calls to a hash function. This is in contrast with previous works, which always involved much more complex operations, such as FFTs [FJR22] or compositions of random permutations [FJR21; BGK+22]. While there is not an optimized implementation of the signature scheme (even if there is an implementation for the improved signature in the next chapter), it is possible to carefully estimate the runtime of all operations using standard benchmarks, making conservative choices when the exact cost is unclear (the calculations are explained in details in Section 5.4.2.1). The conservative choices likely overestimate the real runtime of these operations: of course, the runtimes extrapolated this way ignore other costs such as the cost of copying and allocating memory. Nevertheless, in Banquet, another candidate post-quantum signature scheme using the MPC-in-the-head paradigm, the memory costs were estimated to account for 25% of the total runtime. Therefore, one can expect the extrapolated number to be relatively close to real runtimes with a proper implementation.
For communication, eight sets of parameters are provided. The first four parameters use RSD parameters which guarantee a security reduction to the standard RSD assumption, and one can view them as the main candidate parameters. They correspond respectively to a fast signature (rsd-f), two medium signatures (rsd-m1 and rsd-m2) achieving a reasonable speed/size tradeoff, and a short signature (rsd-s). The last four parameters (arsd-f, arsd-m1, arsd-m2, and arsd-s) use a more aggressive setting of the RSD parameters, where security is reduced instead to a more exotic assumption, namely, the security of RSD when the adversary is allowed to find an *almost regular* solution (with some fixed number of "faulty blocks" allowed). Since this variant has not yet been thoroughly analyzed, these parameters can mainly be viewed as a motivation for future cryptanalysis of variants of RSD with almost-regular solutions.
In Table 5.1 it is possible to find the results of the estimations and the comparison of them to the state-of-the-art in code-based signature schemes. Compared to the best-known code-based signature scheme of [FJR22], the conservative scheme (under standard RSD) achieves significantly smaller signature sizes than their scheme based on syndrome decoding over $\mathbb{F}_2$ (12.52 KB for the fast variant versus 17 KB for Var2f, and 9.69 to 8.55 KB for the shorter variants versus 11.8 KB for Var2s). In terms of runtime, the estimates are significantly faster than their reported runtimes (except rsd-s, which is on par with Var2s), hence the runtimes

should remain competitive with a proper implementation, even if memory costs turn out to be higher than expected. Their most efficient scheme (variants Var3f and Var3s) relies on the conjectured hardness of syndrome decoding assumption over $\mathbb{F}_{256}$, which has been much less investigated.

5.2 Zero-Knowledge Proof for Regular Syndrome Decoding

5.2.1 Template construction

The starting point is the construction of a zero-knowledge proof of knowledge of a solution to an instance of the syndrome decoding problem, using the MPC-in-the-head paradigm. More generally, the presented protocol handles naturally any *syndrome decoding under $\mathbb{N}$-linear constraints* problem for some $\mathbb{N}$-linear constraint $(L, \mathbf{v})$, see Definition 3.5.2. To this end, it will be presented an N-party protocol Π where the parties have shares of a solution $x \in \{0,1\}^K$ to the syndrome decoding problem, and securely output $H \cdot x \bmod 2$ and $L \cdot x$ over $\mathbb{N}$. Given the output of the MPC protocol, the verifier checks:

(1) that the execution (in the prover's head) was carried out honestly (by checking a random subset of $N-1$ views of the parties),

(2) that the two outputs are equal to y and $\mathbf{v}$ respectively.

The high-level intuition of the proposed approach is the following: in MPC protocols, it is typically the case that linear operations are extremely cheap (or even considered as "free") because they can be computed directly over secret values shared using a linear secret sharing scheme (such as additive sharing, or Shamir sharing), without communicating. In turn, one can observe that several variants of the syndrome decoding problem reduce to finding a solution x that satisfies two types of linear constraints: one linear constraint over $\mathbb{F}_2$ (typically, checking that $H \cdot x = y$ given a syndrome decoding instance (H, y)) and one linear constraint over $\mathbb{N}$ (*e.g.* checking that $\langle x, \mathbf{1} \rangle = w$, *i.e.* that the Hamming weight of x is w). Now, an appropriate choice of linear secret sharing scheme can make any one of these two constraints *for free* in Π: if x is additively shared over $\mathbb{F}_2$, then verifying $H \cdot x = y$ is for free, while if x is additively shared over a large enough integer ring $\mathcal{R} = \mathbb{Z}_T$ (such that no overflow occurs when computing $L \cdot x$ over $\mathbb{N}$ for any $x \in \{0,1\}^K$), then verifying $L \cdot x = \mathbf{v}$ is for free.

5.2.1.1 Share conversion. By the above observation, the only missing ingredient to construct Π is a *share conversion* mechanism: a protocol where the parties start with $\mathbb{F}_2$-shares $[\![x]\!]_2$ of x, and securely convert them to $\mathcal{R}$-shares $[\![x]\!]_T$ of x. The next observation is that for any integer ring $\mathbb{Z}_T$, this can be done easily using appropriate *preprocessing material*. Consider the case of a single bit $a \in \{0,1\}$; the parties initially have $\mathbb{F}_2$-shares $[\![a]\!]_2$ of a. Suppose now that the parties receive the $([\![b]\!]_2, [\![b]\!]_T)$ for a random $b \in \{0,1\}$ from a trusted dealer. The parties can locally compute $[\![a \oplus b]\!]_2$ and open the bit $c = a \oplus b$ by broadcasting their shares. Now, since $a = c \oplus b = c + b - 2b$ over $\mathbb{N}$, only two cases may arise:

Table 5.1: Comparison of the signature scheme with other code-based signature schemes from the literature, for 128 bits of security. All timings are in milliseconds. Reported timings are those extracted in [FJR22] from the original publications, using a 3.5 GHz Intel Xeon E3-1240 v5 for Wave, a 2.8 GHz Intel Core i5-7440HQ for Durandal, and a 3.8 GHz Intel Core i7 for [FJR21; FJR22]. The timings are estimated runtimes with the methodology given in Section 5.4.2.1.

Scheme	$\lvert\mathsf{sgn}\rvert$	$\lvert\mathsf{pk}\rvert$	t_{sgn}	Assumption
Wave	2.07 KB	3.2 MB	300	large-weight SD over $\mathbb{F}_3$, $(U, U+V)$-codes indist.
Durandal - I	3.97 KB	14.9 KB	4	Rank SD over $\mathbb{F}_{2^m}$
Durandal - II	4.90 KB	18.2 KB	5	Rank SD over $\mathbb{F}_{2^m}$
LESS-FM - I	9.77 KB	15.2 KB	-	Linear Code Equivalence
LESS-FM - II	206 KB	5.25 KB	-	Perm. Code Equivalence
LESS-FM - III	11.57 KB	10.39 KB	-	Perm. Code Equivalence
[GPS21] - 256	24.0 KB	0.11 KB	-	SD over $\mathbb{F}_{256}$
[GPS21] - 256	19.8 KB	0.12 KB	-	SD over $\mathbb{F}_{1024}$
[FJR21] (fast)	22.6 KB	0.09 KB	13	SD over $\mathbb{F}_2$
[FJR21] (short)	16.0 KB	0.09 KB	62	SD over $\mathbb{F}_2$
[BGK+22] Sig1	23.7 KB	0.1 KB	-	SD over $\mathbb{F}_2$
[BGK+22] Sig2	20.6 KB	0.2 KB	-	(QC)SD over $\mathbb{F}_2$
[FJR22] - Var1f	15.6 KB	0.09 KB	-	SD over $\mathbb{F}_2$
[FJR22] - Var1s	10.9 KB	0.09 KB	-	SD over $\mathbb{F}_2$
[FJR22] - Var2f	17.0 KB	0.09 KB	13	SD over $\mathbb{F}_2$
[FJR22] - Var2s	11.8 KB	0.09 KB	64	SD over $\mathbb{F}_2$
[FJR22] - Var3f	11.5 KB	0.14 KB	6	SD over $\mathbb{F}_{256}$
[FJR22] - Var3s	8.26 KB	0.14 KB	30	SD over $\mathbb{F}_{256}$
Our scheme - rsd-f	12.52 KB	0.09 KB	2.8*	RSD over $\mathbb{F}_2$
Our scheme - rsd-m1	9.69 KB	0.09 KB	17*	RSD over $\mathbb{F}_2$
Our scheme - rsd-m2	9.13 KB	0.09 KB	31*	RSD over $\mathbb{F}_2$
Our scheme - rsd-s	8.55 KB	0.09 KB	65*	RSD over $\mathbb{F}_2$
Our scheme - arsd-f	11.25 KB	0.09 KB	2.4*	f-almost-RSD over $\mathbb{F}_2$
Our scheme - arsd-m1	8.76 KB	0.09 KB	15*	f-almost-RSD over $\mathbb{F}_2$
Our scheme - arsd-m2	8.28 KB	0.09 KB	28*	f-almost-RSD over $\mathbb{F}_2$
Our scheme - arsd-s	7.77 KB	0.09 KB	57*	f-almost-RSD over $\mathbb{F}_2$

* Runtimes obtained using conservative upper bounds on the cycle counts of all operations as described in Section 5.4.2.1, and assuming that the signature is run on one core of a 3.8GHz CPU. I stress that these parameters ignore costs such as copying or allocating memory, and should be seen only as a first-order approximation of the real runtimes.

1. $c = 1$. Then $a = 1 - b$ and so $[\![a]\!]_T = [\![1 - b]\!]_T$.
2. $c = 0$. Then $a = b$ and so $[\![a]\!]_T = [\![b]\!]_T$

Therefore, the parties can compute $[\![a]\!]_T$ as $c \cdot [\![1-b]\!]_T + (1-c) \cdot [\![b]\!]_T$. This extends directly to an integral solution vector x. Hence, in the protocol Π, before the execution, a trusted dealer samples a random vector $r \leftarrow\!\!\$\ \{0,1\}^K$ and distributes $([\![r]\!]_2, [\![r]\!]_T)$ to the parties, where T is such that no overflow can occur when computing $L \cdot x \bmod T$ (to simulate $\mathbb{N}$-linear operations). A similar technique was used previously, in a different context, in [RW19; EGK+20].

5.2.1.2 The use of an MPC protocol with a preprocessing phase Building on this observation, it will be introduced an MPC protocol in the preprocessing model, where the trusted dealer picks a random bitstring r, and distributes $([\![r]\!]_2, [\![r]\!]_T)$ to the parties. On input additive shares of the witness x over $\mathbb{F}_2$, the parties can open $z = r + x$. Using the above observation, all parties can reconstruct shares $[\![x]\!]_T$. Then, any linear equation on x over either $\mathbb{F}_2$ or $\mathbb{Z}_T$ can be verified by opening an appropriate linear combination of the $\mathbb{F}_2$-shares or of the $\mathbb{Z}_T$ shares (this last step does not add any communication when compiling the protocol into a zero-knowledge proof).

At a high level, there are two standard approaches to handling preprocessing material using MPC-in-the-head. The first approach was introduced in [KKW18]. It uses a natural cut-and-choose strategy: the prover plays the role of the trusted dealer, and executes many instances of the preprocessing, committing to all of them. Afterwards, the verifier asks for openings of a subset of all preprocessings, and checks that all opened strings have been honestly constructed. Eventually, the MPC-in-the-head compiler is applied to the protocol, using the unopened committed instances of the preprocessing phases. This approach is very generic, but induces a large overhead, both computationally and communication-wise.

The second approach is tailored to specific types of preprocessing material, such as Beaver triples. It is inspired by the classical sacrificing technique which allows one to check the correctness of a batch of Beaver triples while sacrificing only a few triples. It was used in works such as Banquet [BDK+21], or more recently in [FJR22].

Unfortunately, the first approach induces a large overhead, and the second one is tailored to specific types of preprocessing material. Hence in the discussed context, the structure of the preprocessing material makes it unsuitable. Fortunately, it will be shown that, in the chosen setting, the preprocessing material can be handled *essentially for free*.

The technique works as follows: let the prover compute (and commit to) the preprocessing material $([\![\mathbf{r}]\!]_2, [\![\mathbf{r}]\!]_T)$ himself, but require that the coordinates of $\mathbf{r}$ are *shuffled using a uniformly random permutation* (chosen by the verifier) before being used in the protocol. Crucially, as shown in the analysis, the verifier *never* needs to check that the preprocessing phase was correctly executed (which would induce some overhead): instead, it will be demonstrated that a malicious prover (who does not know a valid witness) cannot find *any* (possibly incorrect) preprocessing material that allows him to pass the verification *with the randomly shuffled material* with high probability.

Fundamentally, the intuition is the following: it is easy for the malicious prover to know values x, x' such that $H \cdot x = y \bmod 2$ and $L \cdot x' = \mathbf{v} \bmod T$. To pass the verification test in the protocol, a malicious prover must therefore fine-tune malicious preprocessing strings

$(\mathbf{s}, \mathbf{t})$ such that the value $\mathbf{z} \odot (\mathbf{1} - \mathbf{t}) + (\mathbf{1} - \mathbf{z}) \odot \mathbf{t}$, computed from $\mathbf{z} = \mathbf{s} \oplus x$ for some x such that $H \cdot x = y \bmod 2$, is equal to a value x' such that $L \cdot x' = \mathbf{v} \bmod T$ (recall that in the honest protocol, the prover should use $\mathbf{s} = \mathbf{t} = \mathbf{r}$). But doing so requires a careful choice of the entries (s_i, t_i): intuitively, the prover needs $s_i = t_i$ whenever $x_i = x'_i$, and $s_i = 1 - t_i$ otherwise. However, when the coordinates of $(\mathbf{s}, \mathbf{t})$ are randomly shuffled, this is not the case with high probability. While the high-level intuition is clear, formalizing it requires particularly delicate combinatorial arguments.
Let (H, y) be an instance of the $\mathbb{N}$-linear syndrome decoding problem with parameters (K, k, w) and feasible $\mathbb{N}$-linear constraint $(L, \mathbf{v})$. Let $x \in \{0,1\}^K$ denote a solution for this instance. One can construct an n-party protocol Π in the preprocessing model, where the parties' inputs are additive shares of x over $\mathbb{F}_2$. The protocol Π securely computes $H \cdot x \bmod 2$ and $L \cdot x$ in the honest-but-curious setting, with corruption of up to $n - 1$ parties. Let $\mathsf{par} \leftarrow (K, k, w, c, H, L)$. The protocol Π_{par} is represented in Protocol 3.

Protocol Π_{par}

Parameters: The protocol Π operates with n parties, denoted $(P_1, \cdots, P_n)$. (K, k, w, c) are four integers with $K > k > w$ and $k > c$. $H \in \{0,1\}^{k \times K}$ and $L \in \mathbb{N}^{c \times K}$ are public matrices. Let $\mathsf{par} \leftarrow (K, k, w, c, H, L)$, and let $T \leftarrow \|L \cdot \mathbf{1}\|_1$. The vector $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$ is forming additive shares $[\![x]\!]_2$ over $\mathbb{F}_2$ of a vector $x \in \{0,1\}^K$.

Inputs: Each party P_i has input $\mathbf{x}_i \in \{0,1\}^K$.

Preprocessing: The trusted dealer samples $\mathbf{r} \leftarrow\!\!\$\ \{0,1\}^K$.
He computes $[\![\mathbf{r}]\!]_2 = (\mathbf{s}_1, \cdots, \mathbf{s}_n) \leftarrow\!\!\$\ \mathsf{Share}_2(\mathbf{r})$ and $[\![\mathbf{r}]\!]_T = (\mathbf{t}_1, \cdots, \mathbf{t}_n) \leftarrow\!\!\$\ \mathsf{Share}_T(\mathbf{r})$, viewing bits as elements of the integer ring $\mathbb{Z}_T$ in the natural way.
He distributes $(\mathbf{s}_i, \mathbf{t}_i)$ to each party P_i.

Online Phase: The protocol proceeds in broadcast rounds.

The parties compute $[\![\mathbf{y}']\!]_2 = H \cdot [\![x]\!]_2$ and $[\![\mathbf{z}]\!]_2 = [\![\mathbf{r}]\!]_2 + [\![x]\!]_2$. All parties open $\mathbf{y}'$ and $\mathbf{z}$.

The parties compute $[\![\mathbf{v}']\!]_T \leftarrow L \cdot (\mathbf{z} \odot [\![\mathbf{1} - \mathbf{r}]\!]_T + (\mathbf{1} - \mathbf{z}) \odot [\![\mathbf{r}]\!]_T)$, viewing $\mathbf{z}$ as a vector over $\mathbb{Z}_T$ in the natural way.

All parties open $\mathbf{v}'$.

Output: The parties output $(\mathbf{y}', \mathbf{v}')$.

Protocol 3: Protocol Π_{par} for securely computing $H \cdot x \bmod 2$ and $L \cdot x$ in the honest-but-curious up to $n - 1$ corruptions.

5.2.1.3 A template zero-knowledge proof. Building upon the above, a template zero-knowledge proof is described in Protocol 4 Looking ahead, the final zero-knowledge proof does

(1) instantiate this template for a carefully chosen flavor of syndrome decoding with $\mathbb{N}$-linear constraints,

(2) introduce many optimizations to the proof, building both upon existing optimizations from previous works, and new optimizations tailored to the chosen setting.

5-round zero-knowledge proof

Parameters. (K, k, w, c) are four integers with $K > k > w$ and $k > c$. $H \in \{0,1\}^{k \times K}$ and $L \in \mathbb{N}^{c \times K}$ are public matrices. $y \in \{0,1\}^k$ and $\mathbf{v} \in \{0,1\}^c$ are public vectors. Let $\mathsf{par} \leftarrow (K, k, w, c, H, L)$, and let $T \leftarrow \|\mathbf{v}\|_1$. Let Commit be a non-interactive commitment scheme.

Inputs. The prover and the verifier have common input par and $(y, \mathbf{v})$, which jointly form an instance of the $\mathbb{N}$-linear syndrome decoding problem. The prover additionally holds a witness $x \in \{0,1\}^K$ which is a solution of the instance: $H \cdot x = y \bmod 2$ and $L \cdot x = \mathbf{v} (= \mathbf{v} \bmod T)$.

Witness Sharing. The prover samples $(\mathbf{x}_1, \cdots, \mathbf{x}_n) \leftarrow\!\!\$\ \mathsf{Share}_2(x)$. Each share $\mathbf{x}_i$ is the input of the virtual party P_i.

Round 1. The prover runs the trusted dealer of Π_{par} and obtains $((\mathbf{s}_1, \cdots, \mathbf{s}_n), (\mathbf{t}_1, \cdots, \mathbf{t}_n)) = (\llbracket \mathbf{r} \rrbracket_2, \llbracket \mathbf{r} \rrbracket_T)$. He computes and sends $c_i \leftarrow\!\!\$\ \mathsf{Commit}(\mathbf{x}_i, \mathbf{s}_i, \mathbf{t}_i)$ for $i = 1$ to n to the verifier.

Round 2. The verifier picks a uniformly random permutation $\pi \leftarrow\!\!\$\ S_K$ and sends it to the prover.

Round 3. The prover runs the online phase of Π_{par} where the parties $(P_1, \cdots, P_n)$ have inputs $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$, using the shuffled preprocessing material $(\llbracket \pi(\mathbf{r}) \rrbracket_2, \llbracket \pi(\mathbf{r}) \rrbracket_T)$. For each party P_i, let $\mathsf{msg}_i = (\mathbf{y}'_i, \mathbf{z}_i, \mathbf{v}'_i)$ denote the list of all messages sent by P_i during the execution. The prover sends $(\mathsf{msg}_1, \cdots, \mathsf{msg}_n)$ to the verifier.

Round 4. The verifier chooses a challenge $d \in [n]$ and sends it to the prover.

Round 5 The prover opens all commitments c_j for $j \neq d$ to the verifier.

Verification. The verifier checks:

- that all commitments were opened correctly;
- that the output of Π_{par} with transcript $(\mathsf{msg}_1, \cdots, \mathsf{msg}_n)$ is equal to $(y, \mathbf{v})$;
- that the messages msg_j sent by P_j are consistent with $(\mathbf{x}_j, \mathbf{s}_j, \mathbf{t}_j)$.

The verifier accepts if and only if all checks succeed.

Protocol 4: Template 5-round zero-knowledge proof for $\mathbb{N}$-linear syndrome decoding using MPC-in-the-head with the protocol Π_{par}

5.2.2 Concrete Instantiation for Regular Syndrome Decoding

With the template 5.2.1.3 construction in mind, which works for all choices of syndrome decoding problem with $\mathbb{N}$-linear constraints, it is now possible to focus on one specific problem. The target of this manuscript is the *regular syndrome decoding problem*, where the linear constraint states that the witness x should be a concatenation of w unit vectors (see Section 3.5.3). The rationale behind this choice stems from the communication complexity of the template zero-knowledge proof from Protocol 4. Intuitively, the communication is dominated by the cost of transmitting the vectors over the ring $\mathbb{Z}_T$ (*i.e.* the $\mathbf{t}_i$ vectors): sending each such vector requires $K \cdot \log T$ bits. Looking ahead, even with proper optimizations, the zero-knowledge proof cannot be competitive with state-of-the-art constructions communication-wise whenever the value of $T = \|L \cdot \mathbf{1}\|_1$ is large.
Typically, for the standard syndrome decoding problem $T = K$, hence the communication involves a $K \cdot \log K$ term, and the overhead is too large (when choosing concrete parameters, K is typically in the thousands, hence $K \log K$ is of the order of a few kilobytes, which becomes a few dozen kilobytes after parallel repetitions). On the other hand, *regular* syndrome decoding appears to minimize this cost: the value of T is only K/w. Hence, by choosing the weight appropriately, one can reduce the value of T.
Moving forward, attention will be directed toward the regular syndrome decoding problem as a primary instantiation of the template. Looking ahead, the goal is to minimize the value of $T = K/w$. Concretely, as shown in Section 5.2.2.1, a standard Chinese remainder theorem trick allows working over the ring $\mathbb{Z}_{T/2}$ instead of $\mathbb{Z}_T$, as long as $\gcd(T/2, 2) = 1$ (*i.e.* $T/2$ is odd; intuitively, this is because the "$\bmod 2$ part" of the equation $L \cdot x = \mathbf{v} \bmod T$ can be obtained at no cost from the $\mathbb{F}_2$-sharing of x, hence it only remains to get $L \cdot x \bmod T/2$ and use the CRT to reconstruct $L \cdot x \bmod T$). The smallest possible value of $T/2$ satisfying the above constraint is $T/2 = 3$, implying $T = K/w = 6$. Therefore, $w = K/6$ is set, which is the smallest value of w that sets the bit-size of the vectors $\mathbf{t}_i$ to its minimal value of $K \cdot \log(T/2) = K \cdot \log 3$.

5.2.2.1 Optimization Starting from the template given in the previews section, it is possible to refine it by using various optimizations: some of those are standard, used *e.g.* in works such as [KKW18; BDK+21; FJR22], and others are new, tailored optimizations.

Using a collision-resistant hash function. The "hash trick" is a standard approach to reduce the communication of public coin zero-knowledge proofs. It builds upon the following observation: in a zero-knowledge proof, the verification equation on a list of messages $(m_1, \cdots, m_\ell)$ often makes the messages *reverse samplable*: the verifier can use the equation to recover what the value of $(m_1, \cdots, m_\ell)$ should be. Whenever this is the case, the communication can be reduced by sending $h = H(m_1, \cdots, m_\ell)$ instead of $(m_1, \cdots, m_\ell)$, where H denotes collision-resistant hash function. The verification proceeds by reconstructing $(m_1, \cdots, m_\ell)$ and checking that $h = H(m_1, \cdots, m_\ell)$, and security follows from the collision resistance of H. As h can be as small as 2λ-bit long, this significantly reduces communication.

Using regular syndrome decoding in systematic form. Without loss of generality, it is possible to set H to be in systematic form, *i.e.* setting $H = [H'|I_k]$, where I_k denotes the identity matrix over $\{0,1\}^{k\times k}$. This strategy was used in the recent code-based signature of [FJR22]. Using H in systematic form, and writing x as $x = (x_1|x_2)$ where $x_1 \in \mathbb{F}_2^{K-k}$, $x_2 \in \mathbb{F}_2^k$, allows to have $Hx = H'x_1 + x_2 = y$. Since the instance (H, y) is public, this implies that the prover need not share x entirely over $\mathbb{F}_2$: it suffices for the prover to share x_1, and all parties can reconstruct $[\![x_2]\!]_2 \leftarrow y \oplus H' \cdot [\![x_1]\!]_2$. Additionally, the parties need not opening $\mathbf{z}$ entirely: denoting $\pi(\mathbf{r}) = (r_1|r_2)$, the parties can open instead $[\![z_1]\!]_2 = [\![x_1 \oplus r_1]\!]_2$ and define $z_2 = H'z_1 \oplus y$. This way, they can reconstruct the complete $\mathbf{z}$ as $\mathbf{z} = (z_1|z_2)$. The rest of the protocol proceeds as before. Following the above considerations, and to simplify notations, from now on the short vector of length $K-k$ in the small field (previously indicated with x_1) is referred to simply as x, and the long vector of size K in the large field is referred to as $\tilde{x}$ (*i.e.* $\tilde{x} = (x|H'x \oplus y)$).

Exploiting the regular structure of x. One can further reduce the size of x using an optimization tailored to the RSD setting: its regular structure allows to divide x into w blocks each of size $T = K/w$. But since each block has exactly one non-zero entry, given the first $T-1$ entries $(b_1, \cdots, b_{T-1})$ of any block, the last entry can be recomputed as $b_T = 1 \oplus \bigoplus_{i=1}^{T-1} b_i$. In the zero-knowledge proof, the prover only shares $T-1$ out of the T bits in each block of x among the virtual parties. Similarly, the size of r_1 and z_1 are reduced by the same factor, since only $T-1$ bits of each block must be masked.
Denoting $x^1, \cdots, x^w$ the blocks of x, therefore:

$$x = \underbrace{\boxed{x_1^1|x_2^1|\cdots|1-\sum_{i=1}^{T-1} x_i^1}}_{x^1} \cdots \underbrace{\boxed{x_1^w|x_2^w|\cdots|1-\sum_{i=1}^{T-1} x_i^w}}_{x^w}$$

Reducing the size of the messages. With the above optimizations, the equation $H\tilde{x} = H'x \oplus x_2 = y$ needs not to be verified anymore: it now holds by construction, as $\tilde{x}$ is defined as $(x|H'x \oplus y)$. This removes the need to include $\mathbf{y}_i$ in the messages msg_i sent by each party P_i; this is in line with previous works, which also observed that linear operations are for free with proper optimizations. The message of each party becomes simply $\mathsf{msg}_i = (\mathbf{z}_i, \mathbf{v}'_i)$. Note that in this concrete instantiation the entries of $\mathbf{v}'_i$ are computed as $\langle \mathbf{1}, \tilde{\mathbf{x}}_\mathbf{i}{}^j \rangle$, where the vectors $\tilde{\mathbf{x}}_\mathbf{i}{}^j$ for $j = 1$ to w are the blocks of P_i's share of the vector $\mathbf{z} \odot (\mathbf{1} - \pi(\mathbf{t})) + (\mathbf{1} - \mathbf{z}) \odot \pi(\mathbf{t})$.

Using the Chinese remainder theorem. In the zero-knowledge proof, verifying any linear equation modulo 2 on the witness $\bar{\mathbf{x}}$ is for free communication-wise. Ultimately, the verifier wants to check that $\langle \bar{\mathbf{x}}^j, \mathbf{1} \rangle = 1 \bmod T$. Setting T to be equal to 2 modulo 4 guarantees that T is even, and $\gcd(T/2, 2) = 1$. Hence, it suffices to work over the integer ring $\mathbb{Z}_{T/2}$ instead of $\mathbb{Z}_T$, to let the verifier check the equation $\langle \bar{\mathbf{x}}^j, \mathbf{1} \rangle = 1 \bmod T/2$ for $j = 1$ to w. Indeed, by the Chinese remainder theorem, together with the relations $\langle \bar{\mathbf{x}}^j, \mathbf{1} \rangle = 1 \bmod 2$ (which can be checked for free), this ensures that $\langle \bar{\mathbf{x}}^j, \mathbf{1} \rangle = 1 \bmod T$ for $j = 1$ to w. This reduces the size of the $\mathbf{t}_i$ vectors from $K \cdot \log T$ to $K \cdot \log(T/2)$. As outlined in Section 5.2.2, if $T = 6$ in the concrete instantiation, hence the protocol has to be executed over the integer ring $\mathbb{Z}_3$, the smallest possible ring satisfying the coprimality constraint.

Compressing share with PRG. Another standard technique from [KKW18] uses a pseudorandom generator to compress all-but-one shares distributed during the input sharing and preprocessing phases. Indeed, writing $[\![\mathbf{x}]\!]_2 = (\mathbf{x_1}, \ldots, \mathbf{x_N})$, then $\mathbf{x_N} = \mathbf{x} - \bigoplus_{i=1}^{N-1} \mathbf{x_i} \bmod 2$. Denoting also $[\![\mathbf{r}]\!]_2 = (\mathbf{s_1}, \ldots, \mathbf{s_N})$ and $[\![\mathbf{r}]\!]_T = (\mathbf{t_1}, \ldots, \mathbf{t_N})$, it holds that $\sum_{i=1}^{N} \mathbf{t_i} = \bigoplus_{i=1}^{N} \mathbf{s_i} \bmod T$, which rewrites to $\mathbf{t}_N = \bigoplus_{i=1}^{N} \mathbf{s_i} - \sum_{i=1}^{N-1} \mathbf{t_i} \bmod T$.
One can compress the description of these shares by giving to each party P_i a $\lambda-$bit seed sd_i and letting each of them apply a pseudorandom generator to sd_i to obtain (pseudo)random shares $\mathbf{x_i}$, $\mathbf{s_i}$ and $\mathbf{t_i}$. All shares of $\mathbf{s}$ can be compressed this way (since $\mathbf{s}$ need just be a random vector), and all but one share of $[\![\mathbf{x}]\!]_2$ and $\mathbf{t}$. The missing shares can be obtained by letting P_N receive an auxiliary string aux_N defined as:

$$\mathsf{aux}_N \leftarrow \left(\mathbf{x} - \bigoplus_{i=1}^{N-1} \mathbf{x_i} \bmod 2, \bigoplus_{i=1}^{N} \mathbf{s_i} - \sum_{i=1}^{N-1} \mathbf{t_i} \bmod T \right).$$

The information shared with each party will be denoted as the *state* of the party. For each P_i for $1 \leq i \leq N-1$, it therefore holds that $\mathsf{state}_i = \mathsf{sd}_i$. The last party P_N has $\mathsf{state}_N = (\mathsf{sd}_N | \mathsf{aux}_N)$: in the online phase of the protocol each party sd_N can be used to randomly generate $\mathbf{s_N}$.

Generating the seeds from a puncturable pseudorandom function. To further reduce the overhead of communicating the seeds, it is possible to apply the standard optimization of generating all seeds from a PPRF (see Section 3.4.5). Concretely, once a master seed sd^* is introduced, it is possible to generate N minor seeds $\mathsf{sd}_1, \cdots, \mathsf{sd}_N$ as the leaves of a binary tree of depth $\log N$, where the two children of each node are computed using a length-doubling pseudorandom generator. This way, revealing all seeds except sd_j requires only sending the seeds on the nodes along the co-path from the root to the j-th leave, which reduces the communication from $\lambda \cdot (N-1)$ to $\lambda \cdot \log n$. Note that due to this optimization, when compiling the proof into a signature, collisions among sd^* for different signatures are likely to appear after $2^{\lambda/2}$ signatures. To avoid this issue, an additional random salt of length 2λ must be used, see Section 5.4.

Using deterministic commitments. As in [KKW18] and other previous works, all committed values are pseudorandom. Therefore, the commitment scheme does not have to be hiding: in the random oracle model, it suffices to instantiate $\mathsf{Commit}(x; r)$ deterministically as $H(x)$ for zero-knowledge to hold.

5.2.2.2 Final Zero Knowledge protocol. The Protocol 5 presents the final zero-knowledge proof of knowledge for a solution to the regular syndrome decoding problem, taking into account all the optimizations outlined above, except the use of deterministic commitments (using deterministic commitments requires using the ROM, which is otherwise not needed for the zero-knowledge proof. Looking ahead, this optimization is still used when compiling the proof to a signature using Fiat-Shamir, since the ROM is used at this stage anyway).

5-round ZKPoK for RSD

Inputs: The prover and the verifier have a matrix $H \in \mathbb{F}_2^{k \times K} = [H'|I_k]$ and a vector $y \in \mathbb{F}_2^k$. The prover also knows a regular vector $\tilde{x} = (x|x_2) \in \mathbb{F}_2^K$ with Hamming weight $\mathsf{HW}(\tilde{x}) = w$ and such that $H\tilde{x} = y$.

Parameters and notations. Let N denote the number of parties. Let x' denote the vector obtained by deleting the T-th bit in each block of x. The action of recomputing x from x', *i.e.* adding a T-th bit at the end of each length-$(T-1)$ block, computed as the opposite of the XOR of all bits of the block, is referred to as "expanding."

Round 1 The prover emulates the preprocessing phase of Π as follows:

1. Chooses a random seed sd^*;
2. Uses sd^* as the root of a depth-$\log N$ full binary tree to produce the leaves $(\mathsf{sd}_i, \sigma_i)$ using a length-doubling PRG for each $i \in [N]$;
3. Use $(\mathsf{sd}_1, \cdots, \mathsf{sd}_{N-1})$ to create pseudorandom shares $(\mathbf{x'_1}, \cdots, \mathbf{x'_{N-1}})$ of x', as well as vectors $(\mathbf{s_i}, \mathbf{t_i}) \in \mathbb{F}_2^{(T-1)\cdot(K-k)/T} \times \mathbb{Z}_{T/2}^K$. Use sd_N to create $\mathbf{s_n}$ as well. Let $\mathbf{x_i}$ denote the vector obtained by "expanding" $\mathbf{x'_i}$ to $K-k$ bits;
4. Let $\mathbf{s'_i}$ denote the value obtained by "expanding" $\mathbf{s_i}$ to a $(K-k)$-bit vector, and let $s' \leftarrow \bigoplus_{i=1}^n \mathbf{s'_i}$. Set $s \leftarrow (s'|H' \cdot s' \oplus y)$. Define
$$\mathsf{aux}_N \leftarrow \left(x' \oplus \bigoplus_{i=1}^{N-1} \mathbf{x_i}, s' - \sum_{i=1}^{N-1} \mathbf{t_i} \bmod T/2 \right);$$
5. Sets $\mathsf{state}_i = \mathsf{sd}_i$ for $1 \le i \le N-1$ and $\mathsf{state}_N = \mathsf{sd}_N || \mathsf{aux}_N$;
6. For each $i \in [N]$ computes $\mathsf{com}_i := \mathsf{Commit}(\mathsf{state}_i, \sigma_i)$;
7. Computes $h := H(\mathsf{com}_1, \cdots, \mathsf{com}_n)$ and sends it to the verifier.

Round 2 The verifier chooses a permutation $\pi \in S_{K-k}$ and sends it to the prover.

Round 3 The prover:

1. Simulates the online phase of the N parties protocol Π using the pairs $(\pi(\mathbf{s_i}), \pi(\mathbf{t_i}))$ as the preprocessing material of the i-th party:
   - for each $i \in [N]$ compute $\mathbf{z'_i} = \mathbf{x'_i} \oplus \pi(\mathbf{s_i})$ getting $[\![z_1]\!]_2 = (\mathbf{z_1}, \cdots, \mathbf{z_N})$ by "expanding" the $\mathbf{z'_i}$'s;
   - Define $z_2 = H' \cdot z_1 \oplus y$ and $z = (z_1|z_2)$;
   - Set $[\![\bar{x}]\!]_{T/2} = (\bar{\mathbf{x_1}}, \ldots, \bar{\mathbf{x_N}})$ where $\bar{\mathbf{x_i}} = z + (1-2z) \odot \pi(\mathbf{t_i}) \bmod T/2$;
   - For each $i \in [N]$ compute:
     - $\bar{\mathbf{w}}_\mathbf{i}^\mathbf{j} = \mathsf{HW}(\bar{\mathbf{x}}_\mathbf{i}^j) \bmod T/2$ for all the blocks $1 \le j \le w$;
     - $\mathsf{msg}_i = (\mathbf{z_i}, (\bar{\mathbf{w}}_\mathbf{i}^\mathbf{j})_{1 \le j \le w})$;

2. Compute $h' = H(\mathsf{msg}_1, \cdots, \mathsf{msg}_n)$;
3. Send z_1 and h' to the verifier. // sending z_1' actually suffices

Round 4 The verifier chooses a challenge $d \in [n]$ and sends it to the prover.
Round 5 The prover sends $(\mathsf{state}_i, \sigma_i)_{i \neq d}$ and com_d.
Verification The verifier checks that everything is correct:

1. Recompute $\bar{\mathsf{com}}_j = \mathsf{Commit}(\mathsf{state}_j, \sigma_j)$ for $j \neq d$;
2. Recompute msg_i for all $i \neq d$ using state_i and z_1;
3. Recompute
$$\bar{\mathsf{msg}}_d = \left(z_1 - \sum_{i \neq d} \mathbf{z_i}, \left(1 - \sum_{i \neq d} \bar{\mathbf{w}}_\mathbf{i}^\mathbf{j} \right)_{1 \leq j \leq w} \right);$$
4. Check if $h = H(\bar{\mathsf{com}}_1, \cdots, \mathsf{com}_d, \cdots, \bar{\mathsf{com}}_N)$;
5. Check if $h' = H(\mathsf{msg}_1, \cdots, \bar{\mathsf{msg}}_d, \cdots, \mathsf{msg}_N)$.

Protocol 5: A five-round zero-knowledge proof of knowledge of a solution to the regular syndrome decoding problem

5.3 Efficiency and Performance Analysis

5.3.1 Communication

The expected communication of the zero-knowledge argument amounts to:

$$4\lambda + \tau \cdot \left(\lambda(\log N + 1) + \left(\frac{2N-1}{N} \right) \frac{T-1}{T} (K-k) + \left(\frac{N-1}{N} \right) K \log_2 T \right) \text{ bits},$$

where hashes are assumed to be 2λ bits long, and commitments are λ bits long, and where τ denotes the number of parallel repetitions of the proof.

5.3.2 Honest-Verifier Zero-Knowledge and Soundness

The completeness of the protocol naturally derives from its definition. In this section, the honest-verifier zero-knowledge and soundness properties are proven.

5.3.2.1 Honest-Verifier Zero-Knowledge HVZK follows from semi-honest security of the N-parties protocol Π. Given a simulator Sim_Π for Π, it is possible to construct an honest-verifier zero-knowledge simulator for the presented protocol as follows:

- Run Sim_Π using the permutation π (which can be computed from the random coins of the adversary V against Π) to simulate the views of parties $P_{i \neq d}$ to obtain $\mathsf{state}_{i \neq d}$

and z. From these informations the simulator computes all the message msg_i, and so $h' = H(\mathsf{msg}_1, \cdots, \mathsf{msg}_N)$, as an honest prover would;

- For all $i \neq d$ chooses uniform σ_i and computes $\mathsf{com}_i = \mathsf{Commit}(\mathsf{state}_i, \sigma_i)$ as an honest prover would. Moreover, computes com_d as a commitment to a 0-string. Hence the simulator can compute $h = H(\mathsf{com}_1, \cdots, \mathsf{com}_N)$;
- The simulator outputs h, h', z and $(\mathsf{state}_i, \sigma_i)_{i \neq d}$, msg_d.

A straightforward hybrid argument shows that transcripts output by the simulator are computationally indistinguishable from transcripts of real executions of the protocol with an honest verifier.

5.3.2.2 Soundness

Theorem 5.3.1. *Let* Commit *be a non-interactive commitment scheme, and H be a collision-resistant hash function. Let* p *be a combinatorial bound (5.3.3) for the Protocol 5. Then the protocol is a* gap *honest-verifier zero-knowledge argument of knowledge for the relation $\mathcal{R}$ such that $((H, y), x) \in \mathcal{R}$ if $H \cdot x = y \bmod 2$ and x is a regular vector of weight w. The* gap *relation $\mathcal{R}'$ is such that $((H, y), x) \in \mathcal{R}'$ if $H \cdot x = y \bmod 2$ and x is an f-weakly valid witness. The soundness error of the proof is at most $\varepsilon = \mathsf{p} + 1/n - \mathsf{p}/n$.*

Proof. Let $\tilde{\mathsf{P}}$ be a prover that manages to generate an accepting proof with probability $\tilde{\varepsilon} > \varepsilon$. It exists an *extractor* which finds a witness x such that $H \cdot x = y$, where x is guaranteed to be a *weakly valid* witness (see Definition 5.3.2). Let R denote the randomness used by $\tilde{\mathsf{P}}$ to generate the commitment h of the first round, and by r a possible realization of R. Let $\mathsf{Succ}_{\tilde{\mathsf{P}}}$ denote the event that $\tilde{\mathsf{P}}$ succeeds in convincing V. By hypothesis

$$\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}] = \tilde{\varepsilon} > \varepsilon = \mathsf{p} + \frac{1}{N} - \frac{\mathsf{p}}{N}.$$

Since the goal is to show that having several accepted transcripts available it is always possible to reconstruct a correct witness, then it is necessary to compute the probability that the malicious prover $\tilde{\mathsf{P}}$ succeeds in convincing the verifier V more than once. As the first round of the protocol is fixed, the events are not independent, so a different argument from the splitting lemma is required to calculate this probability.
Let fix an arbitrary value $\alpha \in \{0, 1\}$ such that $(1 - \alpha)\tilde{\varepsilon} > \varepsilon$, which exists since $\tilde{\varepsilon} > \varepsilon$. Let's say that a realization r of the prover randomness for the first flow is *good* if it holds that

$$\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}} | R = r] \geq (1 - \alpha)\tilde{\varepsilon}.$$

Furthermore, by the Splitting Lemma (see e.g. [FJR22]), $\Pr[R \text{ good} | \mathsf{Succ}_{\tilde{\mathsf{P}}}] \geq \alpha$. Assume now that T_0 is the transcript of a successful execution of the zero-knowledge proof with $\tilde{\mathsf{P}}$. Let r denote the random coin used by $\tilde{\mathsf{P}}$ in the first round, and let d_0 denote the fourth-round

message of the verifier. If r is *good*, then

$$\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R=r] \geq (1-\alpha)\tilde{\varepsilon} > \varepsilon > \frac{1}{N},$$

which implies that there necessarily exists a second successful transcript T_1 with a different fourth-round message $d_1 \neq d_0$. As it will be demonstrated afterward, given (T_0, T_1), it is possible to extract a unique well-defined triplet (x, s, t) (where x is a candidate witness, and (s, t) is the preprocessing material used by $\tilde{\mathsf{P}}$) consistent with both transcripts.

Consistency of (T_0, T_1). Let (π_0, d_0) and (π_1, d_1) be the verifier challenges in the successful transcripts T_0 and T_1 respectively, with $d_0 \neq d_1$. Let us denote $(\mathsf{state}'_{i\neq d_0}, \sigma'_{i\neq d_0}, \mathsf{com}_{d_0})$ and $(\mathsf{state}_{i\neq d_1}, \sigma_{i\neq d_1}, \mathsf{com}_{d_1})$ the rest of the transcripts T_0 and T_1 respectively. Suppose that $\exists i \in [N] \setminus \{d_0, d_1\}$ such that $(\mathsf{state}_i, \sigma_i) \neq (\mathsf{state}'_i, \sigma'_i)$. Then there are two possibilities:

- The committed values are different:

  $$\mathsf{com}_i = \mathsf{Commit}(\mathsf{state}_i, \sigma_i) \neq \mathsf{Commit}(\mathsf{state}'_i, \sigma'_i) = \mathsf{com}'_i.$$

  But since both transcripts from such states are supposed to be accepted, this implies that in particular $h = H(\mathsf{com}_1, \cdots, \mathsf{com}_N)$ and $h = H(\mathsf{com}'_1, \cdots, \mathsf{com}'_N)$ which contradicts the collision resistance of H.

- The committed values are equal:

  $$\mathsf{com}_i = \mathsf{Commit}(\mathsf{state}_i, \sigma_i) = \mathsf{Commit}(\mathsf{state}'_i, \sigma'_i) = \mathsf{com}'_i.$$

  This directly contradicts the binding property of Commit.

Therefore, the states and the randomness are necessarily mutually consistent (that is $\mathsf{state}'_{i\neq d_0,d_1} = \mathsf{state}_{i\neq d_0,d_1}$ and $\sigma'_{i\neq d_0,d_1} = \sigma_{i\neq d_0,d_1}$). Since $d_0 \neq d_1$, they jointly define a unique tuple $(\mathsf{state}_i, \sigma_i)_{i\in[N]}$, from which it is possible to recompute $[\![x]\!]_2 = (\mathbf{x_1}, \ldots, \mathbf{x}_N)$ and $[\![s]\!]_2 = (\mathbf{s_1}, \ldots, \mathbf{s_n})$, $[\![t]\!]_p = (\mathbf{t_1}, \ldots, \mathbf{t_n})$.

The witness x is a valid witness. In this section is showed that if x is a strongly invalid witness, then $\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R=r] \leq \varepsilon$, contradicting the assumption that r is good. Let denote BadPerm the event (defined over the random choice of a permutation π, and for the fixed value of (x, s, t)) that in each block of x', the entries of the blocks sum to 1 modulo $T/2$ (that is, the event $\mathsf{Succ}(x')$), where $x' = \pi(t) + (x \oplus \pi(s)) \odot (\mathbf{1} - 2\pi(t))$. By construction, the extraction procedure guarantees that the extracted candidate witness x has blocks of odd Hamming weight. Therefore, by definition of the combinatorial bound (Definition 5.3.3),

it holds that $\Pr[\mathsf{BadPerm}] \leq \mathsf{p}$. Now,

$$\begin{aligned}\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R=r] &= \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}} \ \wedge \ \mathsf{BadPerm}|R=r] + \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}} \ \wedge \ \neg\mathsf{BadPerm}|R=r] \\ &\leq \mathsf{p} + (1-\mathsf{p}) \cdot \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R=r \ \wedge \ \neg\mathsf{BadPerm}].\end{aligned}$$

Assume for the sake of contradiction that $\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R=r \ \wedge \ \neg\mathsf{BadPerm}] > 1/n$. As before, this implies that given a successful transcript T'_0 with fourth round d'_0, there necessarily exists a second successful transcript T'_1 with the same first three rounds, and a fourth round $d'_1 \neq d'_0$.

By the same argument as above, T'_0 and T'_1 are necessarily consistent, and uniquely define a tuple $(\mathsf{state}'_i, \sigma'_i)_{i\in[N]}$. Furthermore, since $R = r$, meaning that the first flow h' is the same as the first flow h in T_0, T_1, it must holds that $(\mathsf{state}'_i, \sigma'_i)_{i\in[N]} = (\mathsf{state}_i, \sigma_i)_{i\in[N]}$, the states and random coins uniquely defined by (T_0, T_1) (if this is not the case, either a contradiction to the collision-resistance of H or the binding of Commit is obtained, as already shown).

Now, given T'_0, reconstruct the messages: $\mathsf{msg}_i = (\mathbf{z}_i, (\bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{i}})_{j\leq w})$ is computed as $\mathbf{z}_i \leftarrow \mathbf{x}_i \oplus \pi(\mathbf{s}_i)$ and $\bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{i}} = \langle 1, \bar{\mathbf{x}}_{\mathbf{i}}{}^j \rangle$. In these equations, $\mathbf{x}_i, \mathbf{s}_i$ are stretched from sd_i, and $\bar{\mathbf{x}}_{\mathbf{i}}$ is computed as $z + (1-2z) \odot \pi(\mathbf{t}_i)$, with $z = (z_1 | H' z_1 \oplus y)$ and $\mathbf{t}_i$ stretched from sd_i (or computed from aux_n if $i = n$). Define $\mathbf{z}_{d'_0} \leftarrow z_1 - \sum_{i\neq d'_0} \mathbf{z}_i$ and $\bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{d'_0}} \leftarrow 1 - \sum_{i\neq d'_0} \bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{i}}$ for $j = 1$ to w. The remaining tuple, $\bar{\mathsf{msg}}_{d'_0}$, is computed as $(\mathbf{z}_{d'_0}, (\bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{d'_0}})_{j\leq w})$.

Because T'_0 and T'_1 are consistent, all messages msg_i for $i \notin \{d'_0, d'_1\}$ reconstructed from T'_1 must be identical to those reconstructed from T'_0. However, the value $\bar{\mathsf{msg}}_{d'_0}$ is necessarily distinct from the value $\mathsf{msg}_{d'_0}$ reconstructed from T'_1. Indeed, if this was not the case, since $\bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{d'_0}} = 1 - \sum_{i\neq d'_0} \bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{i}}$, this would imply that the values $\bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{i}})_{i\leq n}$ form additive shares of 1 for $j = 1$ to w, implying that the value $\bar{x} = \bigoplus_i \bar{\mathbf{x}}_{\mathbf{i}}$ has blocks of Hamming weights all equal to 1 modulo $T/2$. But since $\bar{x} = \pi(t) + (x \oplus \pi(s)) \odot (\mathbf{1} - 2\pi(t))$ where (x, s, t) are constructed from $(\mathsf{state}'_i, \sigma'_i)_{i\in[N]} = (\mathsf{state}_i, \sigma_i)_{i\in[N]}$, this implies that there would be a contradiction to $\neg\mathsf{BadPerm}$.

Hence, it must be $\mathsf{msg}_{d'_0} \neq \bar{\mathsf{msg}}_{d'_0}$. But since T'_0, T'_1 have the same first three rounds, and in particular the same hash

$$h' = H(\mathsf{msg}_1, \cdots, \bar{\mathsf{msg}}_{d'_0}, \cdots, \mathsf{msg}_n) = H(\mathsf{msg}_1, \cdots, \mathsf{msg}_{d'_0}, \cdots, \bar{\mathsf{msg}}_{d'_1}, \cdots, \mathsf{msg}_n)$$

, a contradiction to the collision-resistance of H has been reached. Hence, assuming the collision-resistance of H, it necessarily holds that $\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R=r \ \wedge \ \neg\mathsf{BadPerm}] \leq 1/n$. Finishing the proof:

$$\begin{aligned}\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R=r] &\leq \mathsf{p} + (1-\mathsf{p}) \cdot \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R=r \ \wedge \ \neg\mathsf{BadPerm}] \\ &\leq \mathsf{p} + (1-\mathsf{p}) \cdot \frac{1}{n} = \varepsilon,\end{aligned}$$

contradicting the assumption that r is good. Therefore, x cannot be a strongly invalid witness.

The extractor. Equipped with the above analysis, an extractor $\mathcal{E}$, which is given rewindable black-box access to a prover $\tilde{\mathsf{P}}$, is defined. Let $N \leftarrow \ln(2)/((1-\alpha)\tilde{\varepsilon} - \varepsilon)$. $\mathcal{E}$ works as follows:

- Run $\tilde{\mathsf{P}}$ and simulate a honest verifier V to get a transcript T_0. Restart until T_0 is a successful transcript.
- Do N times:
  - Run $\tilde{\mathsf{P}}$ with a honest V and the same randomness as in T_0 to get a transcript T_1.
  - If T_1 is a successful transcript with $d_0 \neq d_1$, extract the tuple (x, s, t). If x is a weakly valid witness, output x.

The end of the proof is perfectly identical to the analysis in [FJR22, Appendix F]: given that $\mathcal{E}$ found a first successful transcript T_0, then

$$\begin{aligned}\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}^{T_1} \wedge d_1 \neq d_0 | R \text{ good}] &= \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}^{T_1} | R \text{ good}] - \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}^{T_1} \wedge d_1 = d_0 | R \text{ good}] \\ &\geq (1-\alpha)\tilde{\varepsilon} - 1/n \geq (1-\alpha)\tilde{\varepsilon} - \varepsilon,\end{aligned}$$

hence by definition of N, $\mathcal{E}$ gets a second successful transcript with probability at least $1/2$. From there, the analysis of the expected number of calls $\mathbb{E}[\mathsf{call}]$ of $\mathcal{E}$ to $\tilde{\mathsf{P}}$ is identical to [FJR22, Appendix F] (indeed, the defined extractor is identical, and the zero-knowledge proof has a similar structure):

$$\begin{aligned}&\mathbb{E}[\mathsf{call}] \leq 1 + (1 - \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}]) \cdot \mathbb{E}[\mathsf{call}] + \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}] \cdot (N + (1-\alpha/2) \cdot \mathbb{E}[\mathsf{call}]) \\ \implies &\mathbb{E}[\mathsf{call}] \leq \frac{2}{\alpha\tilde{\varepsilon}} \cdot \left(1 + \tilde{\varepsilon} \cdot \frac{\ln(2)}{(1-\alpha)\tilde{\varepsilon} - \varepsilon}\right),\end{aligned}$$

which gives an expected number of calls $\mathsf{poly}(\lambda)(\lambda, (\tilde{\varepsilon} - \varepsilon)^{-1})$ by setting $\alpha \leftarrow (1 - \varepsilon/\tilde{\varepsilon})/2$ (corresponding to $(1-\alpha)\tilde{\varepsilon} = (\varepsilon + \tilde{\varepsilon})/2$). This concludes the proof. □

5.3.3 Combinatorial Analysis

The discussion so far hinged upon the assumption that when the preprocessing material is randomly shuffled by the verifier, a cheating prover has a very low success probability. More specifically, as shown in Theorem 5.3.1, the soundness error of the new proof system depends on a combinatorial bound p, which bounds the probability that a cheating prover with an incorrect witness x finds preprocessing material (s, t) such that the verifier test with $x, \pi(s), \pi(t)$ passes, over the choice of the random permutation π.

Definition 5.3.1 (informal). *A real* $\mathsf{p} \in (0,1)$ *is a* combinatorial bound *for the template zero-knowledge proof if for every incorrect witness* x*, and every pair* $(\mathbf{s}, \mathbf{t})$*, the probability, over the random choice of a permutation* π*, that* x *satisfies the following equations:*

- $x' = \mathbf{z} \odot (\mathbf{1} - \pi(\mathbf{t})) + (\mathbf{1} - \mathbf{z}) \odot \pi(\mathbf{t})$ *with* $\mathbf{z} = \pi(\mathbf{s}) \oplus x$
- $H \cdot x = y \bmod 2$*,* $L \cdot x = \mathbf{v} \bmod 2$*, and* $L \cdot x' = \mathbf{v} \bmod T/2$

is upper-bounded by p.

Note that the last two equations stem from the use of the gcd trick, where the "mod 2 part" of the equation $L \cdot x = \mathbf{v} \bmod T$ is verified directly on the original shares of x modulo 2, and the remaining equation is checked modulo $T/2$ (assuming that $\gcd(2, T/2) = 1$). Proving a tight combinatorial bound turns out to be a highly non-trivial task. However, this section provides an explicit formula for computing a tight combinatorial bound p in the setting where $T = K/w = 6$ (corresponding to $\mathbb{Z}_{T/2}$ being the smallest ring whose order is coprime with 2; this is the choice that minimizes the communication of the proof). In this setting, a valid witness is a concatenation of w blocks of length 6, each block being a unit vector.
To formally define the combinatorial bound, another definition is necessary.

Definition 5.3.2 (f-Strongly invalid candidate witness). *A vector* $x \in \mathbb{F}_2^K$ *is called a* f-weakly valid witness *if* x *is* almost *a regular vector (in the sense that it differs from a regular vector in at most* f *blocks), or almost an* antiregular *vector. Formally, let* $(x^j)_{j \le w}$ *be the* w *length-*K/w *blocks of* x*. Assume that* K/w *is even. Then* x *is an* f*-weakly valid witness if*

1. $\forall j \le w$, $\mathsf{HW}(x^j) = 1 \bmod 2$*, and*
2. $|\{j \ : \ \mathsf{HW}(x^j) \neq 1\}| \le f$ or $|\{j \ : \ \mathsf{HW}((\mathbf{1} \oplus x)^j) \neq 1\}| \le f$,

where $\mathbf{1} \oplus x$ *is the vector obtained by flipping all bits of* x*. If* x *is not an* f*-weakly valid witness, it is referred to as an* f-strongly invalid *candidate witness.*

Below, set $T \leftarrow K/w$. For simplicity, assume that the parameters are such that w divides K, and that $T = 2 \bmod 4$. Note that this ensures that a block x^j of the candidate witness x has Hamming weight 1 if and only if $\sum_{i=1}^{K/w} x_i^j = 1 \bmod T/2$ and $\sum_{i=1}^{K/w} x_i^j = 1 \bmod 2$.

Definition 5.3.3 (Combinatorial Bound). *Given a vector* $u \in \mathbb{N}^K$ *divided into* w *length-*K/w *blocks* u^j*, let* $\mathsf{Succ}(u)$ *denote the event that* $\sum_{i=1}^{K/w} u_i^j = 1 \bmod T/2$ *for all* $j \le w$*. A* combinatorial bound *for the zero-knowledge proof of 5 with parameters* (K, w) *is a real* $\mathsf{p} = \mathsf{p}(K, w, f) \in (0,1)$ *such that for* any f*-strongly invalid candidate witness* $x \in \mathbb{F}_2^K$ *satisfying* $\forall j \le w$, $\mathsf{HW}(x^j) = 1 \bmod 2$ (i.e., x *still satisfies condition* 1 *of Definition 5.3.2), and for* any *pair of vectors* $(s, t) \in \mathbb{F}_2^K \times \mathbb{Z}_{T/2}^K$,

$$\Pr[\pi \leftarrow\!\!\$\ \mathsf{Perm}_K, x' \leftarrow \pi(t) + (x \oplus \pi(s)) \odot (\mathbf{1} - 2\pi(t)) \ : \ \mathsf{Succ}(x')] \le \mathsf{p}(K, w, f),$$

where Perm_K *denotes the set of all permutations of* $\{1, \cdots, K\}$.

5.3.3.1 Overview - A balls-and-bins analysis.

A key difficulty in the analysis is the need to handle arbitrary choices of the strings $(\mathbf{s}, \mathbf{t})$ chosen by the prover, as well as arbitrary (invalid) witnesses x. In the concrete instantiation, the regular syndrome decoding problem is used, with $T = K/w = 6$ (a choice maximizing efficiency). Therefore, the analysis focuses on this setting. In this case, assume an incorrect witness x is given, represented as a concatenation of w length-T blocks $x^1, \cdots, x^w$. The equation $L \cdot x = \mathbf{v} \bmod 2$ translates to the condition that each block x^j has odd Hamming weight; since $T = 6$, $\mathsf{HW}(x^j) \in \{1, 3, 5\}$ for $j = 1$ to w.
Now fix a position $i \leq K$. The pair $(s_{\pi(i)}, t_{\pi(i)}) \in \mathbb{F}_2 \times \mathbb{F}_3$ "transforms" x_i into x'_i as follows: $x'_i = (x_i \oplus s_{\pi(i)}) \cdot (1 - 2t_{\pi(i)}) + t_{\pi(i)}$. The six elements of $\mathbb{F}_2 \times \mathbb{F}_3$ fall into three categories based on their effect on x_i:

- (Identity) $x'_i = x_i$. This occurs when $s_{\pi(i)} = t_{\pi(i)}$.
- (Flip) $x'_i = 1 \oplus x_i$. This occurs when $t_{\pi(i)} \in \{0, 1\}$ and $s_{\pi(i)} \neq t_{\pi(i)}$.
- (Constant 2) $x'_i = 2$. This occurs when $t_{\pi(i)} = 2$.

Thus, the prover's choice of $(\mathbf{s}, \mathbf{t})$ effectively selects a sequence of $(\mathsf{copy}, \mathsf{flip}, \mathsf{const2})$ operators, which are then randomly shuffled and applied to each bit of the witness x. This can be modeled as a balls-into-bins experiment: the witness x is viewed as a sequence of K bins, where the i-th bin is labeled by x_i. The prover selects K balls, each representing an operator (type-A for copy, type-B for flip, and type-C for const2). The balls are then randomly assigned to bins, and the label of each bin is modified according to the operator applied. The prover succeeds if the sum of the labels in each block of bins is 1 modulo 3 (ensuring each block of x' has Hamming weight 1 modulo 3).
The analysis distinguishes two cases based on the balls chosen by the prover: either at least 60% of the balls are of the same type (this type is said to *dominate* the balls, with the 60% threshold being somewhat arbitrary), or the types are *well-spread* (no type appears more than 60% of the time). Intuitively, these cases correspond to two different "failure modes":

- **Dominant Scenario.** In this case, the best choice for the prover is to select x "very close" to a valid witness (for example, with a single incorrect block) and set almost all balls as type-A balls (type-A being what an honest prover would select). A few type-B balls are then introduced, with the hope that the permutation places the type-B balls exactly within the incorrect blocks of x, thereby correcting them. Alternatively, x can be chosen to be close to an "anti-valid" witness (*i.e.* a valid witness with all its bits flipped), and almost all balls are set as type-B balls to achieve the same effect. Bounding this scenario involves calculating the probability that the incorrect blocks of x receive balls of the dominant types.
- **Well-Spread Scenario.** In the well-spread scenario, each bin receives a ball chosen randomly from the initial set of balls. Since the distribution is well-spread, the label of each bin is mapped to an element of $\{0, 1, 2\}$ with a well-distributed probability mass over the options. To succeed, sufficiently many of the labels must be correctly set (ensuring all blocks have labels summing to $1 \bmod 3$). If the random variables

corresponding to the labels were independent, a Chernoff bound would demonstrate that this occurs with very low probability. Although the labels are not entirely independent, the problem reduces to bounding a hypergeometric distribution, for which strong Chernoff-style bounds are available (see Lemma 5.3.2).

To bound the dominant scenario, a (slightly involved) counting argument is used, enumerating the total number of winning configurations for the prover for each choice of (1) the number of incorrect blocks in x (denoted ℓ), and (2) the number of balls of the dominant type (denoted θ), and dividing this by the total number of configurations. For each choice of (ℓ, θ), this results in an explicit (albeit complex) formula for the bound. It is conjectured that the optimal choice of ℓ, θ is to set $\ell = 1$ and $\theta = K - 1$ (*i.e.*, using a witness with a single incorrect block). Although no proof of this statement is provided, the bound can still be computed explicitly by minimizing the formula over all possible choices of ℓ and θ. For concrete parameters, a Python script[1] is used to compute the bound explicitly. The output of the script confirmed the conjecture for all parameters tested.
In contrast, in the well-spread scenario, the analysis bounds p using a Chernoff-style bound for hypergeometric distribution, which directly provides an explicit and simple formula for computing p in this case. Due to the exponential decay of the bound, it is observed that the well-spread scenario is never advantageous for a malicious prover: the optimal strategy is always to set $(\mathbf{s}, \mathbf{t})$ to fall within the dominant scenario.

5.3.3.2 A Formula for Estimating p

An adversary using an *f-strongly invalid witness*, as defined in Definition 5.3.2, is considered. The adversary may choose preprocessing material $([\![s]\!]_2, [\![t]\!]_3)$ (*i.e.*, s may differ from t). The verifier selects and sends a uniformly random permutation $\pi : [K] \mapsto [K]$. After opening $z = x \oplus \pi(s)$, $[\![x']\!]_3$ is computed as $[\![\pi(t) + z - 2z \odot \pi(t)]\!]_3$. The following lemma provides an explicit formula for estimating p:

Lemma 5.3.1. *Assume x is an f-strongly invalid witness satisfying $\forall j \leq w$, $\mathsf{HW}(x^j) = 1 \bmod 2$. Then:*

$$\mathsf{p} \leq 1 - \min\left\{ \min_{\substack{0.6K \leq \theta < K \\ f < \ell < K/6 - f}} \left(\frac{F(j, K, \ell, \theta)}{\binom{K}{\theta}} \right) \; ; \; 1 - 0.96^K/(2K) \; ; \; (1 - e^{-0.2 \cdot K/6}) \cdot \left(1 - \frac{1}{\binom{K/6}{K/60}} \right) \right\},$$

where

$$F(j, K, \ell, \theta) := \sum_{j=1}^{K/6} (-1)^{j+1} \cdot \sum_{i=0}^{j} 6^i \cdot \binom{\ell}{i} \cdot \binom{K/6 - \ell}{j - i} \cdot \binom{K - 6 \cdot j}{\theta - 6 \cdot j + i}.$$

n the formula above, the first term of $\min$ corresponds to the case where the preprocessing material (s, t) is *mostly honest* (*i.e.* $s_i = t_i$ for most positions) or *mostly dishonest* (*i.e.* $s_i \neq t_i$

[1] https://github.com/ElianaCarozza/Short-Signatures-from-Regular-Syndrome-Decoding-in-the-Head/blob/main/script.py

for most positions), while the second term corresponds to the case where (s, t) is *well spread* (having many positions with $s_i \neq t_i$ and many with $s_i = t_i$).
In the latter case, passing the verifier test requires a high degree of luck with the permutation π: since the honest and dishonest positions in (s, t) are randomly shuffled, they are highly likely to misalign with the invalid witness x. This intuition is formalized by showing that the success event in this case is dominated by an event following a hypergeometric distribution, and standard concentration bounds for hypergeometric distributions are applied.
The former case is more complex. Here, the success probability is bounded for every fixed number ℓ of *incorrect blocks* in the witness x (*i.e.*, blocks whose Hamming weight is not 1), using counting arguments and the inclusion-exclusion principle. The bound is then minimized over all possible choices of ℓ, excluding only the cases $\ell \leq f$ and $\ell \geq K/6 - f$, as these correspond to the witness x being an *f-weakly valid witness*, according to Definition 5.3.2.
It is worth noting that the same analysis provides a bound on the probability of using any witness other than a strictly regular or strictly antiregular vector. However, this bound is insufficient for the intended purpose. Intuitively, a cheating prover can adopt the following strategy: use a witness x that is valid everywhere except on a single block, and select (s, t) to be honest preprocessing material except at a single position i^*. By appropriately setting the value at the incorrect position, the cheating prover passes the verifier test whenever the permutation π aligns i^* with the faulty block of x. This event occurs with probability $6/K$. For typical parameter choices, this quantity falls within the range $[250, 350]$, leading to a failure bound p in the range of $1/300$. This is too high for the intended purpose, as the goal is to achieve small signatures, which requires designing a "one-shot" zero-knowledge proof with low soundness error.

A conjecture. It is conjectured that this naive strategy is the best possible. That is, if the cheating prover holds an f-strongly invalid witness, the best they can do is

1. use a witness with exactly two incorrect blocks,
2. choose (s, t) to be an honest preprocessing material except on two positions (i_0, i_1).

Then, the prover wins if and only if the permutation aligns exactly i_0, i_1 each with one of the two incorrect blocks. In this case, the winning probability of the prover is bounded by $\binom{w}{2}^{-1}$. This conjecture is stated below:

Conjecture 5.3.1. *If x is an f-strongly invalid witness satisfying $\forall j \leq w$, $\mathsf{HW}(x^j) = 1 \bmod 2$, then*

$$\mathsf{p} \leq \binom{w}{f}^{-1}.$$

There is no proof of this conjecture. Nevertheless, for any concrete choice of the parameter K, it is not too hard to compute the concrete bound: a small Python program [2] can be run to explicitly compute the formula of Lemma 5.3.1. When this is done, it is observed that the

[2] https://github.com/ElianaCarozza/Short-Signatures-from-Regular-Syndrome-Decoding-in-the-Head/blob/main/script.py

bound obtained is very close to the bound of the conjecture. For example, computing the formula with $f = 2$ and $K = 1284$ yields a value of p approximately equal to $1/22880$, only slightly looser than the bound of $1/22898$ predicted by the conjecture. Note that naively computing the double sum of products of binomials for all values of ℓ in $[f+1, K/6-f-1]$ would yield very slow runtimes (a few days over a laptop for values of K in the thousands). However, carefully storing some of the binomials and reusing them when appropriate makes the computation significantly faster. Below, the proof of Lemma 5.3.1 is provided.

Notations. In the following, let Fail denote the event, defined over the random sampling of π, that x' does not pass the verifier check; *i.e.*, that $\mathsf{Succ}(x') = 0$. In other words, the event Fail is raised if there is $i \in [1, K/6]$ such that $\sum_{j=6i-5}^{6i} x'_j \neq 1 \bmod 3$. Useful notations are introduced to discuss the type of preprocessing material (s, t) that a cheating prover can use. Observe that a pair $(s_i, t_i) \in \mathbb{F}_2 \times \mathbb{F}_3$ can take six possible values. These pairs are categorized into three types:

Type A. $(s_i, t_i) \in \{(0,0), (1,1)\}$, which are referred to as the copy type,

Type B. $(s_i, t_i) \in \{(0,1), (1,0)\}$, which are referred to as the flip type,

Type C. $(s_i, t_i) \in \{(0,2), (1,2)\}$, which are referred to as the $\mathsf{const2}$ type.

The names $\mathsf{copy}, \mathsf{flip}, \mathsf{const2}$ are explained. It is helpful to understand these categories by viewing a pair (s_i, t_i) as the following *operator* which transforms a bit $u \in \mathbb{F}_2$ into $u' \in \mathbb{F}_3$:

$$f_{s_i,t_i} : u \to t_i + (u \oplus s_i) - 2 \cdot (u \oplus s_i) \cdot t_i \bmod 3.$$

It is easy to check that the six possible pairs (s_i, t_i) correspond only to three possible distinct operators. A type-A pair corresponds to the copy operator, which transforms $u \in \mathbb{F}_2$ into $u \in \mathbb{F}_3$. A type-B pair corresponds to the flip operator, which transforms $u \in \mathbb{F}_2$ into $1 - u \in \mathbb{F}_3$. Eventually, a type-C pair corresponds to the $\mathsf{const2}$ operator, which maps any $u \in \mathbb{F}_2$ to the constant $2 \in \mathbb{F}_3$.
The analysis distinguishes two complementary scenarios for the vectors (s, t) (whose shares form the preprocessing material): either a type is *dominant* among the vector components (*i.e.*, a significant proportion of all pairs (s_i, t_i) are of the same type), or the types are *well-spread*. The cutoff between these two scenarios does not matter much, but for the sake of concreteness, (s, t) is in the *dominant* scenario if there is a type (either A, B, or C) such that more than 60% of all pairs (s_i, t_i) are of this type, and (s, t) is in the *well-spread* scenario otherwise. Each scenario is analyzed separately.

Concentration inequality. A variant of the Chernoff inequality for hypergeometric distributions [3] is used:

[3] Let X be a random hypergeometric variable, with the population size$= N$, number of successes in the population$= K$, and the number of samples drawn without replacement$= n$. The expected value of X is $\mu = \mathbb{E}[X] = n \cdot \frac{K}{N}$. For any $\delta > 0$, $Pr[X \geq (1+\delta)\mu] \leq \exp\left(-\frac{\delta^2 \mu}{2(1+\delta/3)}\right)$ and $\Pr[X \leq (1-\delta)\mu] \leq \exp\left(-\frac{\delta^2 \mu}{2}\right)$.

Lemma 5.3.2. *Suppose there is an urn with N balls, P of which are black. A random sample of n balls is drawn from the urn without replacement. Let $H(N, P, n)$ denote the number of black balls in the sample, and let $q \leftarrow P/N$. Then for any $t \in [0, 1-q]$,*

$$\Pr[H(N, P, n) \geq (q+t) \cdot n] \leq \exp\left(-n \cdot D_{\mathsf{KL}}(q+t||q)\right),$$

where $D_{\mathsf{KL}}(u||v) = u \cdot \ln(u/v) + (1-u) \cdot \ln((1-u)/(1-v))$ denotes the Kullback-Leibler divergence (or relative entropy).

5.3.3.3 The Dominant Scenario It is assumed in this section that more than 60% of all pairs (s_i, t_i) are of the same type.

Case 1: type A is dominant. First, assume that the dominant type is A: at least 60% of all pairs are of the copy type. Let $\theta \geq 0.6 \cdot K$ denote the number of type-A pairs in $(\llbracket s \rrbracket, \llbracket t \rrbracket_3)$, *i.e.*, the number of coordinates $i \in [K]$ such that $s_i = t_i$.

Lemma 5.3.3. *Assume that x is a strongly invalid witness satisfying $\forall j \leq w$, $\mathsf{HW}(x^j) = 1 \bmod 2$, and that the number of type-A pairs in (s, t) is $\theta \geq 0.6K$. It holds that*

$$\Pr_{\pi}[\mathsf{Fail}] \geq \frac{\min_{f<\ell<K/6-f}\left(\sum_{j=1}^{K/6}(-1)^{j+1} \cdot \sum_{i=0}^{j} 6^i \cdot \binom{\ell}{i} \cdot \binom{K/6-\ell}{j-i} \cdot \binom{K-6\cdot j}{\theta-6\cdot j+i}\right)}{\binom{K}{\theta}}.$$

To prove Lemma 5.3.3, some useful notations are introduced. Recall that x is divided into $K/6$ blocks x^j of length 6 (with $j \leq K/6$), where $x^j = (x_{6j-5}, \cdots, x_{6j})$. Because of the condition that x satisfies $\forall j \leq w$, $\mathsf{HW}(x^j) = 1 \bmod 2$ (which is guaranteed by construction in the proof), each x^j has Hamming weight either 1, 3, or 5. For every block x^j, x^j is referred to as a *honest block* if $\mathsf{HW}(x^j) = 1$, and as a *dishonest block* otherwise (observe that if all the x^j are honest blocks, then x is a valid witness).

Proof. To bound the probability of the event Fail, it is reformulated as a balls-and-bins game. Each bin corresponds to an entry of the vector x, and each ball to a pair (s_i, t_i) from the preprocessing material. For the proof of Lemma 5.3.3, two main types of balls are distinguished: type-A balls and non-type-A balls. The experiment then becomes: K balls are randomly thrown into K bins, where θ balls are type-A balls, and $K-\theta$ balls are non-type-A balls. The value of θ is adversarially chosen but satisfies $\theta \geq 0.6 \cdot K$, and each ball ends up in exactly one bin. Each bin i is labeled with the corresponding bit x_i of x. The bins are divided into $K/6$ blocks $B_1, \cdots, B_{K/6}$ of six bins. When a ball falls into a bin i, it changes its label x_i into a new label x'_i according to the operator: the label remains the same for type-A balls (copy type), it is "flipped" for type-B balls (flip type), and it is replaced by 2 for type-C balls (const2 type).
A block of bins B_j is called honest if its bins are labeled with an honest block x_j of the witness x; otherwise, it is a dishonest block of bins. Let ℓ denote the number of honest blocks of bins. For simplicity, it is assumed that the honest blocks are B_1 to B_ℓ, and the remaining

blocks are the dishonest blocks; this assumption is made without loss of generality.
The following events $E_j = E_j^{(\ell)}$ are defined for $j = 1$ to $K/6$ (the superscript ℓ is omitted for readability): for $j = 1$ to ℓ, E_j is the event that exactly five type-A balls end up in the block B_j, and for $j = \ell + 1$ to $K/6$, E_j is the event that exactly six type-A balls end up in the block B_j. The rationale behind the choice of the E_j is as follows:

- if any E_j happens, then the event Fail is triggered,
- when the proportion of type-A balls is large, it is likely that one such event occurs.

Claim 5.3.1.

$$\Pr[\mathsf{Fail}] \geq \Pr\left[\bigcup_{j=1}^{K/6} E_j\right].$$

Proof. It is shown that $\cup_{j=1}^{K/6} E_j \subset \mathsf{Fail}$. Fix any $j \leq K/6$. Assume first that $j \leq \ell$, then E_j is the event that exactly five type-A balls fell into B_j. As B_j is an honest block, it has five bins labeled with 0, and one bin labeled with 1. There are two possibilities:

- Either the five type-A balls fell in the five 0-labeled bins. Then, the labels of these five bins remain 0, and the label of the remaining 1-labeled bin is either flipped to 0 (if the remaining ball is a type-B ball) or set to 2 (if the remaining ball is a type-C ball). Therefore, the Hamming weight of the new labels $\mathsf{HW}(x'_j)$ is either 0 or 2: in both cases, it is not equal to 1 modulo 3, hence it fails the verifier check.
- Otherwise, four 0-labeled bins and the 1-labeled bin have their label unchanged, and the remaining 0-labeled bin is either set to 1 (type-B ball) or to 2 (type-C ball). In this case, $\mathsf{HW}(x'_j) \in \{2, 3\}$, hence $\mathsf{HW}(x'_j) \neq 1 \bmod 3$.

Hence, whenever E_j happens, the second verifier check fails, which proves that $E_j \subset \mathsf{Fail}$ for $j = 1$ to ℓ. Assume now that $j > \ell$: E_j is the event that exactly 6 type-A balls fell into B_j, and B_j is a dishonest block (*i.e.*, $\mathsf{HW}(x_j) \in \{3, 5\}$). Since the six type-A balls leave the labels of all six bins unchanged, the Hamming weight of B_j's labels remains 3 or 5, and neither value is equal to 1 modulo 3, causing the second verifier check to fail again. This proves that $E_j \subset \mathsf{Fail}$ for $j = \ell + 1$ to K as well, which concludes the claim. □ □

It remains to bound $\Pr\left[\cup_{j=1}^{K/6} E_j\right]$. This is done through standard combinatorial arguments: the space of all possible configurations of the experiment is considered, without distinguishing type-B and type-C balls (*i.e.*, configurations are counted by only looking at whether each bin contains a type-A or a non-type-A ball). There are exactly $\binom{K}{\theta}$ possible configurations; the number of configurations where one of the events E_j happens is now counted. By the

inclusion-exclusion principle:

$$\left|\bigcup_{j=1}^{K/6} E_j\right| = \sum_{j=1}^{K/6}(-1)^{j+1} \sum_{S\subset[1,K/6],\ |S|=j} \left|\bigcap_{k\in S} E_k\right|.$$

To bound the terms $|\cap_{k\in S} E_k|$, a distinction is made between the E_k with $k \leq \ell$ and those with $k > \ell$. Fix an integer $i \in [0, j]$. Among all possible subsets $S \subset [1, K/6]$ of size $|S| = j$, there are exactly $\binom{\ell}{i} \cdot \binom{K/6-\ell}{j-i}$ subsets S such that $|S \cap [1, \ell]| = i$. Fix any such subset $S = \{k_1, \cdots, k_i, k_{i+1}, \cdots, k_j\}$, and consider the set of configurations $|E_{k_1} \cap \cdots \cap E_{k_j}|$. Picking a configuration in this set amounts to:

- filling all the (dishonest) blocks of bins $B_{k_{i+1}} \cdots B_{k_j}$ with type-A balls (using $6(j-i)$ type-A balls),
- choosing one bin in each (honest) block $B_{k_1} \cdots B_{k_i}$ which does *not* receive a type-A ball,
- filling all remaining bins in $B_{k_1} \cdots B_{k_i}$ with type-A balls (using $5i$ type-A balls),
- choosing a configuration of the $\theta - 6(j-i) - 5i = \theta - 6j + i$ remaining type-A balls among the $K - 6j$ remaining bins.

There are 6^i ways to pick one bin in each of the i blocks $B_{k_1} \cdots B_{k_i}$, and $\binom{K-6j}{\theta-6j+i}$ ways to pick a configuration of the remaining type-A balls among the remaining bins. Therefore,

$$|E_{k_1} \cap \cdots \cap E_{k_j}| = 6^i \cdot \binom{K-6j}{\theta-6j+i}.$$

Hence, by summing over all subsets S of size j with i elements below ℓ, for $i = 0$ to j, it holds

$$\sum_{S\subset[1,K/6],\ |S|=j} \left|\bigcap_{k\in S} E_k\right| = \sum_{i=0}^{j} \binom{\ell}{i} \cdot \binom{K/6-\ell}{j-i} \cdot 6^i \cdot \binom{K-6j}{\theta-6j+i},$$

and therefore

$$\left|\bigcup_{j=1}^{K/6} E_j\right| = \sum_{j=1}^{K/6}(-1)^{j+1} \sum_{i=0}^{j} \binom{\ell}{i} \cdot \binom{K/6-\ell}{j-i} \cdot 6^i \cdot \binom{K-6j}{\theta-6j+i}.$$

Now, by the previous claim, $\Pr[\mathsf{Fail}] \geq \Pr[\cup_{j=1}^{K/6} E_j] = |\cup_{j=1}^{K/6} E_j| / \binom{K}{\theta}$. Since the number ℓ of honest blocks of bins is chosen by the adversary, the failure probability is minimized over all possible values of ℓ, excluding only the cases $\ell \geq K/6 - f$ and $\ell \leq f$, since x is assumed to be an f-strongly invalid witness (Definition 5.3.2). This concludes the proof of Lemma 5.3.3. □

Case 2: type B is dominant. The case where at least 60% of all pairs are of the flip type is now addressed. This case is handled immediately through a simple reduction to Case 1. First, observe that by definition, a vector x is an f-strongly invalid witness if and only if $\mathbf{1} \oplus x$ is: this follows from Definition 5.3.2, and from the fact that $\mathsf{HW}(x^j) = \mathsf{HW}((\mathbf{1} \oplus x)^j)$ for any j since the block length is even. Now, fix any f-strongly invalid witness x. Observe that the probability $\Pr_\pi[\mathsf{Fail}]$ remains identical if

(1) x is replaced by $\mathbf{1} \oplus x$ (*i.e.*, all the bits of x are flipped),

(2) all type B balls are replaced by type A balls and vice-versa (leaving all type C balls untouched).

In other words, flipping all bits of x and exchanging the roles of the copy and flip operators leaves the experiment unchanged (the behavior of const2 operators being unaffected by this change). But this brings the analysis back to the setting where type A is dominant, with an f-strongly invalid witness $\mathbf{1} \oplus x$, and the bound of Lemma 5.3.3 applies.

Case 3: type C is dominant. It remains to address the case where at least 60% of all pairs are of the const2 type. As the fraction of type C balls approaches 1, it is easy to see that $\Pr[\mathsf{Fail}]$ approaches 1 as well, since whenever six type C balls fall into a block of six bins, the event Fail is raised (as $6 \cdot 2 = 0 \neq 1 \bmod 3$). Therefore, without loss of generality, it is assumed that the fraction of type C balls is exactly 0.6. It is shown that in this case,

$$\Pr_\pi[\mathsf{Fail}] \geq 1 - \frac{0.96^K}{2K}.$$

To simplify the analysis, a different distribution is first considered where all bins receive a type A, B, or C ball with the same respective probabilities. This setting is referred to as the *simplified setting* and $\mathsf{Fail}^{\mathsf{s}}$, $\mathsf{Win}^{\mathsf{s}}$ denote the events that the adversary fails or wins in this setting. Now, fix a block b of six bins, and let hw denote the Hamming weight of the labels of the block. For $i = 4, 5, 6$, denote iC the event that exactly i type C balls fall into the block, and let p denote the probability that a type A ball falls into a given bin, conditioned on the event that this bin does not receive a type C ball (then $1 - p$ is the probability that a type B ball falls into the bin, under the same condition). Let $\mathsf{Win}^{\mathsf{s}}_b$ denote the event that the adversary wins for block b (*i.e.*, after placing balls in the bins, the labels sum to $1 \bmod 3$). The probability that the adversary wins conditioned on iC balls falling into the block b is bounded. First, $\Pr[\mathsf{Win}^{\mathsf{s}}_b \mid 6C] = 0$ (since $6 \cdot 2 \neq 1 \bmod 3$). Second,

$$\Pr[\mathsf{Win}^{\mathsf{s}}_b \mid 5C] = (1 - \mathsf{hw}/6)p + (1 - p) \cdot \mathsf{hw}/6 = (1 - \mathsf{hw}/3) \cdot p + \mathsf{hw}/6.$$

Above, $1 - \mathsf{hw}/6$ is the probability that the non-type C bin is a 0-labeled bin: the adversary wins either if a type A ball falls into a 0-labeled bin, or if a type B ball falls into a 1-labeled bin (since $5 \cdot 2 + 0 = 10 = 1 \bmod 3$). Eventually,

$$\Pr[\mathsf{Win}^{\mathsf{s}}_b \mid 4C] = (1-\mathsf{hw}/6)(1-\mathsf{hw}/5)\cdot(1-p)^2+\mathsf{hw}(6-\mathsf{hw})\cdot p(1-p)/15+\mathsf{hw}(\mathsf{hw}-1)\cdot p^2/30.$$

Above, $(1 - \mathsf{hw}/6)(1 - \mathsf{hw}/5) \cdot (1 - p)^2$ is the probability that the two non-type C balls are type B balls and they both fall into 0-labeled bins, $\mathsf{hw}(6 - \mathsf{hw}) \cdot p(1 - p)/15$ is the

probability that the two non-type C balls are a type A and a type B ball and they fall into a 1-labeled and a 0-labeled bin respectively, and $\mathsf{hw}(\mathsf{hw}-1)\cdot p^2/30$ is the probability that the two non-type C balls are type A balls and they both fall into 1-labeled bins. Since $4\cdot 2+2\cdot 1$ is the only way to get $1 \bmod 3$ with 4 type C balls, this covers all winning events for the adversary. Eventually,

$$\begin{aligned}\Pr[\mathsf{Fail}_b^{\mathsf{s}}] &\geq \Pr[\mathsf{Fail}_b^{\mathsf{s}} \wedge 4C] + \Pr[\mathsf{Fail}_b^{\mathsf{s}} \wedge 5C] + \Pr[\mathsf{Fail}_b^{\mathsf{s}} \wedge 6C] \\ &= \Pr[\mathsf{Fail}_b^{\mathsf{s}} \mid 4C]\cdot\Pr[4C] + \Pr[\mathsf{Fail}_b^{\mathsf{s}} \mid 5C]\cdot\Pr[5C] + \Pr[\mathsf{Fail}_b^{\mathsf{s}} \mid 6C]\cdot\Pr[6C] \\ &= \Pr[\mathsf{Fail}_b^{\mathsf{s}} \mid 4C]\cdot\binom{6}{4}\cdot 0.6^4\cdot 0.4^2 + \Pr[\mathsf{Fail}_b^{\mathsf{s}} \mid 5C]\cdot 6\cdot 0.6^5\cdot 0.4 + \Pr[\mathsf{Fail}_b^{\mathsf{s}} \mid 6C]\cdot 0.6^6.\end{aligned}$$

Therefore, for each possible value of the Hamming weight $\mathsf{hw} \in \{0,\cdots,6\}$ of the block of bins, plugging the formulas for $\Pr[\mathsf{Fail}_b^{\mathsf{s}} \mid iC] = 1 - \Pr[\mathsf{Win}_b^{\mathsf{s}} \mid iC]$ for $i = 4,5,6$ above yields a degree-2 polynomial Q_{hw} in p. Let $(\alpha_{\mathsf{hw}}, \beta_{\mathsf{hw}}, \gamma_{\mathsf{hw}})$ denote the coefficients of this polynomial: $Q_{\mathsf{hw}}(p) = \alpha_{\mathsf{hw}}\cdot p^2 + \beta_{\mathsf{hw}}, \cdot p + \gamma_{\mathsf{hw}}$. For each value of $\mathsf{hw} \in \{0,\cdots,6\}$, $Q_{\mathsf{hw}}(p)$ is minimized for $p \in [0,1]$:

- $Q_0(p)$ is minimized at $p = 0$, and $Q_0(0) = 729/3125$,
- $Q_1(p)$ is minimized at $p = 0$, and $Q_0(0) = 4779/15625$,
- $Q_2(p)$ is minimized at $p = 0$, and $Q_0(0) = 5589/15625$,
- $Q_3(p)$ is minimized at $p = 0$ or 1, and $Q_0(0) = 5832/15625$.

Eventually, $Q_4(p)$, $Q_5(p)$, and $Q_6(p)$ are minimized at $p = 1$, and the minima are the same as $Q_2(p)$, $Q_1(p)$, and $Q_0(p)$ respectively (this stems from the fact that the roles of A balls and B balls are symmetrical when 0-labeled bins are exchanged with 1-labeled bins, which changes hw to $6-\mathsf{hw}$).
Overall, the global minimum over (hw, p) is reached at $Q_0(0) = Q_6(1) = 729/3125$. This yields the following bound:

$$\Pr[\mathsf{Fail}_b^{\mathsf{s}}] \geq 729/3125 = 0.23328.$$

Then, across all $K/6$ blocks of bins, the following bound is obtained:

$$\Pr[\mathsf{Fail}^{\mathsf{s}}] \geq 1 - (1-0.23328)^{K/6} \leq 1 - 0.77^{K/6} \leq 0.96^K.$$

Recall that this bound holds in the simplified setting where each bin is assigned a ball whose type is *independently* sampled from $\{A,B,C\}$ with probabilities $p_A = 0.4\cdot p$, $p_B = 0.4\cdot(1-p)$, and $p_C = 0.6$ (that is, a Bernoulli-style distribution is used). Now, let K_A, K_B, K_C denote the random variables that count the number of bins with type A, B, and C balls

respectively. Observe that

$$\begin{aligned}\Pr[\mathsf{Win}^{\mathsf{s}}] &\geq \Pr[\mathsf{Win}^{\mathsf{s}} \wedge (K_A = 0.4pK \wedge K_C = 0.6K)] \\ &= \Pr[\mathsf{Win}^{\mathsf{s}} \mid K_A = 0.4pK, K_C = 0.6K] \cdot \Pr[K_A = 0.4pK \wedge K_C = 0.6K] \\ &\geq \Pr[\mathsf{Win}^{\mathsf{s}} \mid K_A = 0.4pK, K_C = 0.6K] \cdot \Pr[K_A = 0.4pK] \cdot \Pr[K_C = 0.6K].\end{aligned}$$

Furthermore, the probability $\Pr[\mathsf{Win}^{\mathsf{s}} \mid K_A = 0.4pK, K_C = 0.6K]$ corresponds exactly to the event that the adversary wins in the original setting, where a fixed pool of balls (with $0.4pK$ type A, $0.4(1-p)K$ type B, and $0.6K$ type C balls) is randomly permuted and placed into the bins. That is,

$$\Pr[\mathsf{Win}^{\mathsf{s}} \mid K_A = 0.4pK, K_C = 0.6K] = \Pr[\mathsf{Win}^{\mathsf{s}}],$$

Hence

$$\Pr[\mathsf{Win}] \leq \frac{\Pr[\mathsf{Win}^{\mathsf{s}}]}{\Pr[K_A = 0.4pK] \cdot \Pr[K_C = 0.6K]} \leq \frac{0.77^{K/6}}{\Pr[K_A = 0.4pK] \cdot \Pr[K_C = 0.6K]}.$$

To finish the proof, it remains to lower bound $\Pr[K_A = 0.4pK]$ and $\Pr[K_C = 0.6K]$. Since K_A and K_C are both binomial random variables of respective means $0.4pK$ and $0.6K$, this can be done straightforwardly: let K_X be any binomial random variable in $\{0, \cdots, K\}$ with mean μ. Then

$$\begin{aligned}\Pr[K_X = \mu] &= \binom{K}{\mu} \cdot (\mu/K)^{\mu} \cdot (1-\mu/K)^{K-\mu} \\ &= \binom{K}{\mu} \cdot e^{-h(\mu/K)\cdot K},\end{aligned}$$

where $h : x \to -x \ln x - (1-x)\ln(1-x)$ is the binary entropy function. Furthermore, by a standard application of the Stirling inequality[4],

$$\Pr[K_X = \mu] = \binom{K}{\mu} \cdot e^{-h(\mu/K)\cdot K} \geq \sqrt{\frac{K}{8\mu(K-\mu)}} \geq \frac{1}{\sqrt{2K}},$$

where the last inequality is obtained by setting $\mu = K/2$, since it minimizes the expression. Plugging this in the bound on $\Pr[\mathsf{Win}]$, it follows that

$$\Pr[\mathsf{Win}] \leq \frac{0.96^K}{2K},$$

which concludes the analysis of the dominant scenario.

5.3.3.4 The Well-Spread Scenario The same balls-and-bins formalism as in the previous section is used again. In this section, it is assumed that the preprocessing material $([\![s]\!]_2, [\![t]\!]_3)$

[4]This formulation of Stirling's inequality can be found for example in [Gal68].

contains strictly less than 60% of pairs (s_i, t_i) of the most represented type. It is shown that if this is the case, no matter the witness x used by the prover, the event Fail is extremely likely to happen. The core intuition is the following: fix any block B of six bins, and fix any choice of type-A, type-B, and type-C balls for the first five bins. Let y denote the sum modulo 3 of the new labels in these five bins. If the types of balls are sufficiently well-spread, then the distribution of possible new labels for the leftover bin is also well-spread across the set $\{0, 1, 2\}$. For example, in the most extreme case where the unused balls have an equal proportion of each of the three types after a ball is randomly picked and thrown in the leftover bin, the new label of the bin is uniformly random over $\{0, 1, 2\}$, independently of its original label. Therefore, in the well-spread scenario, the new label of the leftover bin has a noticeable probability of being distinct from $1 - y \bmod 3$, in which case a failure event is raised.
The above intuition shows that each block is likely to cause a failure in this scenario. While the events of a block causing a failure are not independent across different blocks, when the total number of balls and bins is large enough, their mutual influence appears very limited, and the event that *at least one* of the blocks causes a failure is expected to quickly become overwhelming. Below, it is proven that this intuition is indeed correct.

Lemma 5.3.4. *Assume that x is a strongly invalid witness satisfying $\forall j \leq w$, $\mathsf{HW}(x^j) = 1 \bmod 2$. If the preprocessing material $(\llbracket s \rrbracket_2, \llbracket t \rrbracket_3)$ contains strictly less than 60% of pairs (s_i, t_i) of the most represented type, it holds that*

$$\Pr_{\pi}[\mathsf{Fail}] \geq (1 - e^{-0.2 \cdot K/6}) \cdot \left(1 - 1/\binom{K/6}{K/60}\right).$$

Proof. Fix one bin in each block (the *selected bins*). Randomly throwing the balls into all bins is equivalent to the following experiment: a uniformly random size-$K/6$ subset of all balls (the *selected balls*) is first picked. The remaining balls are thrown randomly among the remaining bins, and the selected balls randomly among the selected bins. The probability of a failure event is bounded by proving two claims:

- with high probability, the three types of balls (A, B, and C) remain relatively *well-spread* among the randomly selected balls,
- when the selected balls are well-spread, a failure event is extremely likely to happen, for any setting of the remaining balls into the remaining bins.

Claim 5.3.2. *Let S be a uniformly random size-$K/6$ subset of $[1, K]$. Let p denote the probability, over the random choice of S, that the set $\{(s_i, t_i) \ : \ i \in S\}$ contains more than 90% of pairs of any given type. Then*

$$p < \exp\left(-0.2 \cdot K/6\right).$$

Proof. Let θ/K denote the proportion of the most frequent type among all pairs (s_i, t_i); by

assumption, $\theta/K < 0.6$. The balls of the most frequent type are viewed as "black balls" (in the case of ties, the exact choice does not matter). The claim follows by applying the Chernoff inequality for the hypergeometric distribution given in Lemma 5.3.2, using $q = 0.6$ and $t = 0.3$. □ □

Claim 5.3.3. *Fix a set S of $K/6$ selected balls such that the set $\{(s_i, t_i) \ : \ i \in S\}$ does not contain more than 90% of pairs of any given type. Fix $K/6$ selected bins (one in each block). Fix an arbitrary repartition of the remaining balls into the remaining bins. Then the probability, taken over the random assignment of the $K/6$ selected balls to the $K/6$ selected bins, that no failure event happens is bounded by*

$$\Pr[\textit{no failure}] \leq \frac{1}{\binom{K/6}{K/60}}.$$

Proof. Let $1/3 \leq \theta/K < 0.9$ denote the proportion of the most common type among all selected balls. "Black balls" refer to the balls of this type, and "white balls" to the remaining balls. For any block, and any assignment of non-selected balls among the five non-selected bins of this block, there is exactly one type of ball (among A, B, and C) such that assigning this ball to the selected bin of the block does not lead to a failure: if y is the sum (modulo 3) of the new labels in the non-selected bins, this is the type which changes the label of the selected bin to $1 - y \bmod 3$. Imagine now that each selected bin is painted as follows: if the type that does not cause a failure in this block is that of the black balls, it is colored in black; otherwise, it is colored in white.
Two cases can occur: either the number of black balls among the selected balls is not equal to the number of black selected bins. In this case, no assignment can put a ball of the right color in all selected bins, and $\Pr[\text{no failure}] = 0$. Otherwise, assume that exactly $\theta/6$ selected bins are black. To avoid a failure event, the $\theta/6$ balls that end up in the $\theta/6$ black selected bins must be exactly all the black balls. There are $\binom{K/6}{\theta/6}$ ways to choose which balls will end up in the black bins among the $K/6$ selected balls, and only one non-losing configuration, hence

$$\Pr[\text{no failure}] \leq \frac{1}{\binom{K/6}{\theta/6}}.$$

From here, the claim follows using the fact that $\binom{K/6}{\theta/6} \geq \binom{K/6}{0.9 \cdot K/6} = \binom{K/6}{0.1 \cdot K/6}$ since $0.1 < \theta/K < 0.9$. □ □

Equipped with the two claims, the proof of Lemma 5.3.4 follows almost immediately: fix $K/6$ selected bins, one per block, and let S denote the indices of the balls that end up in these bins. Define S as well-spread if no type represents more than 90% of the types of the

balls in S. Then,

$$\begin{aligned}\Pr[\mathsf{Fail}] &\geq \Pr[S \text{ is well-spread}] \cdot \Pr[\mathsf{Fail} \mid S \text{ is well-spread}] \\ &\geq (1 - e^{-0.2K/6}) \cdot \left(1 - \frac{1}{\binom{K/6}{K/60}}\right),\end{aligned}$$

which concludes the proof. □ □

This concludes the proof of Lemma 5.3.1.

5.4 Signature Scheme Construction

In this section, the 5-round protocol described in Protocol 5 is turned into a signature scheme using the Fiat-Shamir transform following the steps presented in the protocol 4.3.5.1. As in previous works, a salt $\mathsf{salt} \in \{0,1\}^{2\lambda}$ is used to avoid $2^{\lambda/2}$-query attack resulting from collisions between seeds. Taking into account the forgery attack presented by Kales and Zaverucha [KZ20a] against the signature schemes obtained by applying the Fiat-Shamir transform to 5-round protocols and adapting it to the actual working context yields a forgery cost of

$$\mathrm{cost}_{\mathrm{forge}} = \min_{\tau_1,\tau_2:\tau_1+\tau_2=\tau} \left\{ \frac{1}{\sum_{i=\tau_1}^{\tau} \binom{\tau}{i} \mathsf{p}^i (1-\mathsf{p})^{\tau-i}} + N^{\tau_2} \right\} \tag{5.1}$$

5.4.1 Description of the Signature Scheme

For the signature scheme central to this chapter, the key generation algorithm randomly samples a syndrome decoding instance (H, y) with solution x as described in Protocol 6.

Key generation algorithm

Inputs: A security parameter λ.

1. Randomly chooses a $\mathsf{sd} \leftarrow \{0,1\}^{\lambda}$;
2. Using a pseudorandom generator with sd to obtain a regular vector $x \in \mathbb{F}_2^K$ with $\mathsf{HW}(x) = w$ and a matrix H;
3. Compute $y = Hx$;
4. Set $\mathsf{pk} = (H, y)$ and $\mathsf{sk} = (H, y, x)$.

Protocol 6: Key generation algorithm

Given a secret key sk and a message m, the signing algorithm is described in Protocol 7.

Signing algorithm of the signature scheme

Inputs: A secret key $\mathsf{sk} = (H, y, x)$ and a message $m \in \{0,1\}^*$.
Sample a random $\mathsf{salt} \in \{0,1\}^{2\lambda}$.
Phase 1 For each iteration $e \in [\tau]$

1. Choose a random seed $\mathsf{sd}_e \leftarrow \{0,1\}^{\lambda}$;
2. Use sd_e and salt as input of a pseudorandom generator to produce sd_i^e for each $i \in [N]$;
3. Compute aux_N^e;
4. Set $\mathsf{state}_i^e = \mathsf{sd}_i^e$ for $1 \leq i \leq N-1$ and $\mathsf{state}_N^e = \mathsf{sd}_N^e || \mathsf{aux}_N^e$;
5. Use all the states to create, through a pseudorandom generator:
   - $[\![x^e]\!]_2 = (\mathbf{x_1^e}, \ldots, \mathbf{x}_N^e)$;
   - $s = [\![r_1^e]\!]_2 = (\mathbf{s_1^e}, \ldots, \mathbf{s_n^e})$;
   - $t = [\![r^e]\!]_q = (\mathbf{t_1^e}, \ldots, \mathbf{t_n^e})$;
6. For each $i \in [N]$ computes $\mathsf{com}_i^e := H_0(\mathsf{salt}, i, \mathsf{state}_i^e)$.

Phase 2

1. Compute $h_1 = H_1(m, \mathsf{salt}, \mathsf{com}_1^1, \cdots, \mathsf{com}_N^1, \cdots, \mathsf{com}_\tau^1, \cdots, \mathsf{com}_N^\tau)$;
2. Obtain $\pi_{\{e\in\tau\}}^e \in S_{K-k}$ via a pseudorandom generator using h_1.

Phase 3 For each iteration $e \in [\tau]$

1. Each party P_i computes $\mathbf{z_i^e} = \mathbf{x_i^e} \oplus \pi(\mathbf{s_i^e})$;
2. The parties get $[\![z_1^e]\!]_2 = (\mathbf{z_1^e}, \cdots, \mathbf{z_N^e})$ and set $[\![z_2^e]\!]_2 = H'[\![z_1^e]\!]_2 \oplus y$ and so $z^e = (z_1^e | z_2^e)$;
3. Obtain $[\![\bar{x^e}]\!]_q = (\bar{\mathbf{x_1^e}}, \ldots, \bar{\mathbf{x_N^e}})$ where $\bar{\mathbf{x_i^e}} = z^e + (1 - 2z^e) * \pi(\mathbf{t_i^e})$;
4. For each $j \in [N]$ compute:
   - $\bar{\mathbf{w}}_\mathbf{i}^{\mathbf{j,e}} = \langle 1, \bar{\mathbf{x}}_\mathbf{i}^{j,e} \rangle$ for all the blocks $1 \leq j \leq w$;
   - $\mathsf{msg}_i^e = \left(\mathbf{z_i^e}, \left\{ \bar{\mathbf{w}}_\mathbf{i}^{\mathbf{j,e}} \right\}_{1 \leq j \leq w} \right)$.

Phase 4

1. Compute $h_2 = H_2(m, \mathsf{salt}, h_1, \mathsf{msg}_1^1, \cdots, \mathsf{msg}_n^1, \cdots, \mathsf{msg}_1^\tau, \cdots, \mathsf{msg}_n^\tau)$;
2. Obtain $d_{\{e\in\tau\}}^e \in [N]$ via a pseudorandom generator using h_2.

Phase 5 Output the signature $\sigma = \mathsf{salt} | h_1 | h_2 | (\mathsf{state}_{i\neq d}^e | \mathsf{com}_{d^e}^e)_{\{e\in\tau\}}$.

Protocol 7: Signing algorithm of the signature scheme

Given a public key pk, a message m and a signature σ, the verification algorithm is described in Protocol 8.

Verification algorithm of the signature scheme

Inputs: A public key $\mathsf{pk} = (H, y)$, a message $m \in \{0,1\}^*$ and a signature σ.

1. Split the signature as follows
$$\sigma = \mathsf{salt}|h_1|h_2|(\mathsf{state}^e_{i\neq d}|\mathsf{com}^e_{d^e})_{\{e\in\tau\}};$$
2. Recompute $\pi^e_{\{e\in\tau\}} \in S_{K-k}$ via a pseudorandom generator using h_1;
3. Recompute $d^e_{\{e\in\tau\}} \in [N]$ via a pseudorandom generator using h_2;
4. For each iteration $e \in [\tau]$
   - For each $i \neq d$ recompute $\bar{\mathsf{com}}^e_i = H_0(\mathsf{salt}, i, \mathsf{state}^e_i)$;
   - Use all the states, except $\mathsf{state}^e_{d^e}$, to simulate the Phase 3 of the signing algorithm for all parties but the d^e-th, obtaining $\mathsf{msg}^e_{i\neq d^e}$;
   - Compute
$$\bar{\mathsf{msg}}^e_{d^e} = \left(z^e_1 - \sum_{i\neq d} \mathbf{z}^{\mathbf{e}}_{\mathbf{i}}, \left\{ 1 - \sum_{i\neq d} \bar{\mathbf{w}}^{\mathbf{j},\mathbf{e}}_{\mathbf{i}} \right\}_{1\leq j\leq w} \right);$$
5. Check if $h_1 = H_1(m, \mathsf{salt}, \mathsf{com}^1_1, \cdots, \mathsf{com}^1_N, \cdots, \mathsf{com}^\tau_1, \cdots, \mathsf{com}^\tau_N)$;
6. Check if $h_2 = H_2(m, \mathsf{salt}, h_1, \mathsf{msg}^1_1, \cdots, \mathsf{msg}^1_n, \cdots, \mathsf{msg}^\tau_1, \cdots, \mathsf{msg}^\tau_n)$;
7. Output ACCEPT if both conditions are satisfied.

Protocol 8: Verification algorithm of the signature scheme

Theorem 5.4.1. *Suppose the* PRG *used is $(t, \epsilon_{\mathsf{PRG}})$-secure and any adversary running in time t has at most an advantage ϵ_{SD} against the underlying d-split syndrome decoding problem. Model the hash functions H_0, H_1, H_2 as random oracles with an output of length 2λ-bit. Then chosen-message adversary against the signature scheme 6, running in time t, making q_s signing queries, and making q_0, q_1, q_2 queries, respectively, to the random oracles, succeeds in outputting a valid forgery with probability*

$$\Pr[\text{Forge}] \leq \frac{(q_0 + \tau N_s)^2}{2 \cdot 2^{2\lambda}} + \frac{q_s\,(q_s + q_0 + q_1 + q_2)}{2^{2\lambda}} + q_s \cdot \tau \cdot \epsilon_{\mathsf{PRG}} + \epsilon_{\mathsf{SD}} + \Pr[X + Y = \tau] \quad (5.2)$$

where $\epsilon = \mathsf{p} + \frac{1}{N} - \frac{\mathsf{p}}{N}$, with p given by Lemma 5.3.1, $X = \max_{\alpha\in Q_1}\{X_\alpha\}$ and $Y = \max_{\beta\in Q_2}\{Y_\beta\}$ with $X_\alpha \sim \mathsf{Binomial}(\tau, \mathsf{p})$ and $Y_\beta \sim \mathsf{Binomial}\left(\tau - X, \frac{1}{N}\right)$ where Q_1 and Q_2 are sets of all queries to oracles H_1 and H_2.

The proof of Theorem 5.4.1 follows directly from the standard analysis of Fiat-Shamir-based signatures from the 5-round identification protocol. It is identical to the proof of Theorem 5 in [FJR22], and it is omitted here.

5.4.2 Parameters Selection Process

This section focuses on the selection of the parameters for the zero-knowledge argument system of Section 5.2.2 and the signature scheme of Section 5.4.1. Let f be the number of faulty blocks (of Hamming weight 3 or 5) allowed in the witness extracted from a cheating prover. Looking ahead, f is chosen as the smallest value that minimizes τ, the number of repetitions of the underlying zero-knowledge argument, which has a strong impact on the size of the signature. Given a candidate value f, the selection of the parameters (K, k, w) proceeds as outlined below. The reader should remind that, to work over the smallest possible field $\mathbb{F}_3$ in the zero-knowledge proof, the weight is forced to be $w = K/6$ to get a blocksize 6. Also, the target bit-security is set to $\lambda = 128$.

Choosing k. As explained in Section 5.5.4, the dimension k is chosen such that even when allowing $f > 0$ faulty blocks in the zero-knowledge proof, the assumption underlying the unforgeability of the signature remains equivalent to the standard RSD assumption. Concretely, this is achieved by setting k to

$$k \leftarrow \left\lceil \log_2 \left(\sum_{i=0}^{f} 6^{w-i} \cdot \binom{w}{i} \cdot 26^i \right) \right\rceil + b \cdot \lambda,$$

with $b = 1$. A second choice of parameters is additionally considered, in which the constant $b = 0$ is used in the above equation. This second choice of parameters corresponds to the f-almost-RSD uniqueness bound, the threshold where the number of almost-regular solutions becomes close to 1. This setting should intuitively lead to the hardest instance of the almost-RSD problem. However, it does not reduce anymore to the standard RSD problem, since a random RSD instance might have irregular (but almost-regular) solutions.

Choosing K. Having chosen k (as a function of $w = K/6$), it is necessary now to focus on the other dimension K. Here, the use of the attacks described in Sections 5.5.2 and 5.5.3 allows to select the smallest K such that, when setting k as above, λ bits of security against all attacks are achieved. Even if the approximate birthday paradox attack (Section 5.5.3) is always the most efficient attack, by a significant margin, it relies upon the assumption that approximate collisions can be found in linear time, and no such linear-time algorithm is known as of today.

Computing p. Equipped with a candidate instance (K, k, w) for a number f of faulty blocks, the use of the formula of Lemma 5.3.1 allows to compute a bound p on the probability that a malicious prover can use an incorrect witness (with at least $f + 1$ faulty blocks) in the first part of the zero-knowledge proof. More precisely, since computing p exactly using the Python code [5] takes a few hours of computation, the value of p is initially determined based

[5] https://github.com/ElianaCarozza/Short-Signatures-from-Regular-Syndrome-Decoding-in-the-Head/blob/main/script.py

on the prediction from Conjecture 5.3.1 (which aligns with all exact calculations tested using the formula). Subsequently, after finalizing all parameter choices, the correctness of the final bound p is verified by applying the explicit formula, ensuring the computation is performed only once.

Computing τ. The number of repetitions τ for the zero-knowledge argument and the signature scheme is computed. At this stage, the parameter selection varies depending on the specific case:

Zero-knowledge argument. For the zero-knowledge argument, τ is computed as the smallest value such that $\varepsilon^\tau \leq 2^{-\lambda}$, where $\varepsilon = 1/n + \mathsf{p} \cdot (1 - 1/n)$, n being the number of parties. Here, there is no optimal choice of f. Instead, f is a tradeoff: choosing $f = 0$ guarantees that the zero-knowledge argument achieves standard soundness (with no gap) but makes ε higher. A larger f reduces p, hence ε, but introduces a gap in soundness. In any case, as soon as $\mathsf{p} \ll 1/n$, it holds that $\varepsilon \approx 1/n$. In practice, using $f = 1$ already leads to $\mathsf{p} < 5 \cdot 10^{-5}$, which is much smaller than any reasonable value of $1/n$ (since increasing n to such values would blow up computation). Hence, the only reasonable choices are $f = 0$ (for standard soundness) and $f = 1$ (for optimal efficiency).

Signature scheme. The signature scheme is obtained by compiling the zero-knowledge argument using Fiat-Shamir. Due to the compilation of a 5-round zero-knowledge proof, the attack by Kales and Zaverucha [KZ20a] applies, necessitating the choice of τ according to Equation 5.1. This significantly alters the optimal selection, as it is no longer valid that any value of $\mathsf{p} \ll 1/n$ automatically results in the smallest possible τ. Instead, by the convexity of Equation 5.1, the smallest achievable τ is $\tau_{\mathsf{ZK}} + 1$, where τ_{ZK} represents the optimal value of τ for the zero-knowledge argument (*i.e.* the smallest value satisfying $\varepsilon^{\tau_{\mathsf{ZK}}} \leq 2^{-\lambda}$). The approach involves computing τ using Equation 5.1 for a given candidate value of f. If $\tau > \tau_{\mathsf{ZK}} + 1$, the value of f is incremented by 1, and the entire process is repeated, including selecting new parameters K and k, recomputing p, and so forth. After a few iterations, the process converges to the smallest number f of faulty blocks that minimizes the resulting value of τ.

Choosing N. The selection of the number of parties N is addressed independently of the other parameters. Increasing N always reduces communication costs by decreasing the soundness error, but it also increases computation, which scales linearly with N. Following the strategy outlined in Banquet [BDK+21], N is chosen as a power of two, aiming for a signing time comparable to that of prior works (on a standard laptop) to ensure a fair comparison. All parameters (K, k, w, f, τ) are then computed, and N is reduced to the smallest value that maintains λ bits of security.

5.4.2.1 Runtime estimations.

Eventually, it remains to estimate the runtime of the signature and verification algorithms of the proposed signature scheme. Since a full-fledged implementation of the signature scheme doesn't exist, existing benchmarks are used to conservatively estimate the runtime of the scheme. The following implementation choices are considered:

- The tree-based PRG is implemented with fixed-key AES [6]. This is the standard and most efficient way to implement such PRGs over machines with hardware support for AES [01].
- The commitment scheme is implemented with fixed-key AES when committing to short values (λ bits), and with SHAKE when committing to larger values.
- The hash function is instantiated with SHAKE.

For fixed-key AES operations, the estimated runtime using hardware instructions is 1.3 cycles/byte [MSY21]. For SHAKE, the runtime strongly depends on a machine. However, according to the ECRYPT benchmarkings[7], on one core of a modern laptop, the cost of hashing long messages ranges from 5 to 8 cycles/byte (8 cycles/byte are used in the estimations, to stay on the conservative side). The analysis also includes counting XOR operations (where XORing two 64-bit machine words requires one cycle) and mod-3 operations. The latter are more challenging to estimate without a concrete implementation. However, their contribution to the overall cost is relatively small. Even a conservative estimation with up to an order of magnitude overhead compared to XOR operations has minimal impact on the overall runtime. An order of magnitude of overhead is assumed in the estimations to remain conservative. When converting cycles to runtime, a 3.8 GHz processor is assumed, consistent with the setup in previous work [FJR22], to facilitate a direct comparison with their results, which are most relevant to this context.
Of course, the above estimations ignore additional costs such as allocating or copying memory, and should therefore only be seen as a rough approximation of the timings that an optimized implementation could get. For comparison, in the Banquet signature scheme [BDK+21], another candidate post-quantum signature scheme based on the MPC-in-the-head paradigm, 25% of the runtime of their optimized implementation was spent on allocating and copying memory, and 75% on the actual (arithmetic and cryptographic) operations.

5.4.2.2 Results. Two settings were considered: a conservative setting, where the underlying assumption reduces to the standard RSD assumption, and an aggressive setting, where the parameters rely on the conjectured hardness of the f-almost-RSD assumption. All numbers are reported in Table 5.1. Depending on the chosen setting, the following parameters were obtained:

Conservative setting (standard RSD). The optimal choice of the number f of faulty blocks is $f = 12$. Given this f, K is set to 1842, k to 1017, and w to 307. The target is 128 bits of security against all known attacks, assuming conservatively that approximate birthday collisions can be found in linear time to estimate the cost of the most efficient attack. In this parameter range, the solution to the random RSD instance is the only 12-almost-regular

[6] A follow-up, discussed in the next chapter, explains that this instantiation was incorrect: as in other works presenting signature candidates, the use of a seed was missing in the AES instantiation, despite being a necessary condition to ensure security. The next chapter provides formal results to properly instantiate the tree-based PRG with fixed-key AES and achieve full security.

[7] https://bench.cr.yp.to/results-hash.html

solution except with probability 2^{-128}; hence, 12-almost-RSD reduces to standard RSD. With these parameters, three values of n were considered. Each time, n was first set to a power of two, the optimal value of τ computed, and then n reduced to the smallest value that still works for this value of τ.

- Setting 1 – fast signature (rsd-f): $\tau = 18$, $n = 193$. In this setting, the signature size is 12.52 KB. The runtime estimated using the described methodology is 2.7ms.
- Setting 2 – medium signature 1 (rsd-m1): $\tau = 13$, $n = 1723$. In this setting, the signature size is 9.69 KB. The runtime estimated using the described methodology is 17ms.
- Setting 3 – medium signature 2 (rsd-m2): $\tau = 12$, $n = 3391$. In this setting, the signature size is 9.13 KB. The runtime estimated using the described methodology is 31ms.
- Setting 4 – short signature 2 (rsd-s): $\tau = 11$, $n = 7644$. In this setting, the signature size is 8.55 KB. The runtime estimated using the described methodology is 65ms.

Aggressive setting (f-almost-RSD). In this setting, k is set at the f-almost-RSD uniqueness bound (the threshold above which the number of f-almost-regular solutions approaches 1). In this setting, there might be additional almost-regular solutions beyond the regular solution x for a random RSD instance; hence, f-almost-RSD does not reduce directly to the standard RSD assumption. This assumption is considered plausible but exotic, and its potential impact on the parameters is investigated. The conservative parameters are viewed as the main choice of parameters. The aggressive parameters yield noticeable improvements in signature size and runtime, which could motivate further cryptanalysis of this exotic variant. Four parameter settings are provided, comparable to the conservative settings, using the optimal value $f = 13$ and the same numbers n of parties as above. In this setting, $K = 1530$, $k = 757$, and $w = 255$.

5.5 Cryptanalysis of RSD

A careful reader might have noticed an apparent issue in the previous analysis: assume that a cheating prover uses an *antiregular* witness x (*i.e.*, a vector such that $x \oplus \mathbf{1}$ is regular), and only type-B balls (*i.e.* the pairs (s_i, t_i) are such that $s_i = 1 - t_i$). Then it passes the verifier checks exactly as an honest prover would: the antiregular vector x still has blocks of odd Hamming weight, and for any choice of π, x' is now equal to $\mathbf{1} \oplus x$: that is, a regular vector. Concretely, this means that the zero-knowledge proof is not a proof of knowledge of a regular solution, but rather a proof of knowledge of either a regular *or an antiregular* solution. Nevertheless, when building a signature scheme, this is not an issue: it simply implies that unforgeability relies instead on the hardness of finding a regular or antiregular solution to an RSD instance. But it is a folklore observation that this variant of RSD does reduce to the standard RSD problem, with only a factor 2 loss in the success probability, hence this does not harm security.

This approach is pushed even further. The bound p obtained by the previous analysis is essentially tight, but remains relatively high for the intended purpose. Concretely, fixing a value of $K \approx 1500$ (this is roughly to the range of parameter choices), the result is $\mathsf{p} \approx 1/250$. This bound is met when the prover uses a witness which is regular almost everywhere, with at most one exceptional block, where it has Hamming weight 3 or 5 (or the antiregular version of that). In this case, the prover builds $(\mathbf{s}, \mathbf{t})$ honestly, except on a single position (s_i, t_i), where $s_i = 1 - t_i$. Then, with probability $1/250$, the permutation aligns i with the unique faulty block (there are 250 blocks in total), and the (s_i, t_i) pair "corrects" the faulty block, passing the verifier checks. Even though a $1/250$ bound is not too bad, in this context it largely dominates the soundness error of the proof. This stems from the fact that the protocol has extremely low computational costs, hence the number n of virtual parties can be set much higher than in previous works, e.g., $n = 1024$ or $n = 2048$, while still achieving comparable computational costs. In this high-n setting, the goal is to achieve a soundness error close to the best possible value of $1/n$, to minimize the number of parallel repetitions (hence reducing communication).

To get around this limitation, almost-regular witnesses (or almost-antiregular witnesses) are allowed. Concretely, the soundness of the zero-knowledge proof is relaxed to guarantee only that a successful cheating prover must at least know an almost-regular (or almost-antiregular) witness, *i.e.*, a witness whose blocks all have weight 1 except one, which might have weight 1, 3, or 5. This form of zero-knowledge proof with a gap between the language of honest witnesses and the language of extracted witnesses is not uncommon in the literature. In particular, it is similar in spirit to the notion of *soundness slack* in some lattice-based zero-knowledge proofs, where a witness is a vector with small entries, and an extracted witness can have much larger entries [CDX+16]. By using this relaxation, the bound p improves by (essentially) a quadratic factor: a cheating prover must now cheat on (at least) *two* positions (s_i, t_i), and hope that both align with the (at least) two incorrect blocks of x. Concretely, using $K \approx 1500$, the combinatorial analysis gives $\mathsf{p} \approx 3 \cdot 10^{-5}$ in this setting, which becomes a vanishing component of the soundness error (dominated by the $1/n$ term). When building a signature scheme from this relaxed zero-knowledge proof, the Fiat-Shamir transform is applied to a 5-round protocol, and the number of repetitions is adjusted to account for the attack of [KZ20a]. For a bound of p as above, this severely harms efficiency. Following the strategy of [FJR22], the problem is avoided by making p much smaller. Concretely, denoting τ_{ZK} the smallest integer such that $(1/n + \mathsf{p} \cdot (1 - 1/n))^{\tau_{\mathsf{ZK}}} \leq 2^{-\lambda}$, the optimal number of repetitions which can be hoped for in the signature scheme is $\tau = \tau_{\mathsf{ZK}} + 1$. Therefore, denoting f the number of faulty blocks in the witness, f is set to be the smallest value such that the resulting bound p yields $\tau = \tau_{\mathsf{ZK}} + 1$, hence achieving the optimal number of repetitions. At this stage, the unforgeability of the signature now reduces to the hardness of finding either an almost-regular or an almost-antiregular solution to an RSD problem (with up to f faulty blocks), which seems quite exotic (though it remains in itself a plausible assumption). For the sake of relying only on the well-established RSD assumption, parameters are set such that, except with $2^{-\lambda}$ probability, a random RSD instance does not in fact have any almost-regular or almost-antiregular solution (with up to f faulty blocks) on top of the original solution. This implies that, for this choice of parameters, this "f-almost-RSD" assumption is in fact equivalent to the RSD assumption (with essentially no

loss in the reduction).
This section gives a fully detailed overview of the security of RSD, exploring also the relation between the RSD problem and the almost-RSD one.
Specifically, a precise description is provided of how RSD relates to the standard syndrome decoding assumption, depending on the parameters (K, k, w). Indeed, although this problem has been used and analyzed in the past, the parameter setting adopted here differs from previous works. Specifically:

- In the work of [AFS03], which introduced the assumption, and its follow-ups [FGS07; MDC+11; BLP+11], the goal was to construct a collision-resistant hash function. Consequently, the parameters were chosen such that the mapping $x \to H \cdot x$ is highly non-injective. In other words, in their parameter setting, an instance (H, y) always has (usually exponentially) many solutions. This, in turn, enabled powerful generalized birthday attacks (GBA), resulting in rather large parameters. In contrast, the setting adopted here relies only on the one-wayness of $x \to H \cdot x$ and does not require the mapping to be compressive.

- In the recent line of work on pseudorandom correlation generators [BCG+18; BCG+19b; BCG+19a; BCG+20b; YWL+20; WYK+21; RS21; CRR21; BCG+22], the regular syndrome decoding problem is typically used in a very specific setting: it relies on the *extremely low noise* regime while allowing the dimension K to be huge, typically of the order of 2^{20} to 2^{30}. This is the opposite of the setting adopted here, where K strongly impacts efficiency (hence, the smaller K, the better), and a high noise regime is used to reduce the block size, further increasing efficiency.

- The work of [HOS+18] also relies on the regular syndrome decoding problem, and their parameter setting was found to be the closest to the one adopted here (though the paper handles a relatively wide variety of parameter settings). A section of [HOS+18] is devoted to the analysis of RSD, and this section served as a starting point. However, a few mistakes and imprecisions were identified in that analysis.

Throughout this chapter, an "RSD uniqueness bound" is concretely defined, analogous to the Gilbert-Varshamov (GV) bound for standard syndrome decoding, and it is shown that:

- above the GV bound, RSD is always easier than SD;

- below the RSD uniqueness bound, RSD becomes *harder* than SD;

- in the intermediate zone between the two bounds, the hardness of the two problems is not directly comparable.

The chosen parameters fall within this gray zone and correspond to a setting where a random RSD instance does not have additional f-almost-regular solutions with high probability, ensuring a tight reduction to the standard RSD assumption even when such relaxed solutions are allowed.
Existing attacks on RSD are also reviewed, and in most cases revisited and improved to exploit the structure of the RSD problem more effectively, resulting in significant speedups.

A new attack is ultimately designed, outperforming all previous attacks. This attack is not fully explicit and relies on an approximate birthday paradox search (*i.e.*, finding an almost-collision between items of two lists). To remain conservative when choosing concrete parameters, it is assumed that this approximate birthday paradox can be solved in time linear to the list size. While it is unclear how such a fast approximate collision search might be performed, it is not implausible that such an algorithm could exist, prompting the decision to err on the side of caution. Identifying such an algorithm is considered an interesting open problem.

5.5.1 Uniqueness Bound for Regular Syndrome Decoding

The Gilbert-Varshamov (GV) bound is defined as the largest d_{GV} such that $\sum_{i=0}^{\mathsf{d}_{\mathsf{GV}}-1} \binom{K}{i} \leq 2^k$. The GV bound establishes a weight threshold for the injective setting of the syndrome decoding problem when the constraint on the solution x is of the form $\mathsf{HW}(x) \leq w$. Specifically, when $w \leq \mathsf{d}_{\mathsf{GV}}$, the pair $(H, y = H \cdot x)$ uniquely determines x with high probability over the choice of a random sparse vector x. When additional structure is imposed on the noise, the number of possible preimages is restricted, altering the threshold. For instance, under an exact weight constraint $\mathsf{HW}(x) = w$ (the standard setting for syndrome decoding), the uniqueness threshold is reached when $\binom{K}{w} \leq 2^k$. In the RSD setting, the uniqueness threshold is achieved when $(K/w)^w \leq 2^k$.

Expected number of solutions. When sampling a random instance $(H, y = H \cdot x)$ of the RSD problem, the expected number of solutions is given by

$$1 + \mathop{\mathbb{E}}_{H,x}\left[\left|\{x' \ : \ H \cdot x' = H \cdot x \ \wedge \ x' \text{ regular}\}\right|\right] = 1 + \sum_{\substack{x' \neq x \\ x' \text{ regular}}} \Pr_{H,x}[H \cdot x' = H \cdot x]$$
$$= 1 + \frac{1}{2^k} \cdot \left((K/w)^w - 1\right),$$

where the last inequality follows from the standard fact that for any pair of vectors (x, x'), $\Pr_H[H \cdot x' = H \cdot x] = 1/2^k$. Therefore, when $((K/w)^w - 1)/2^k$ is very small, the solution x is unique with high probability; when it gets larger, the average number of solutions increases.

Intuition. In the case of the standard syndrome decoding problem, it is well-known that the hardness of the problem is maximized around the GV bound. Similarly, the natural intuition here is that the RSD problem is the hardest when $(K/w)^w \approx 2^k$. In slightly more details, the "highly injective" setting (where $(K/w)^w \ll 2^k$) cannot be harder than the threshold setting, since the adversary can always delete equations, transforming the instance (H, y) into a new instance (H', y') by removing rows of H and entries of y. As long as the adversary maintains $(K/w)^w < 2^k$, it remains likely that the solution to the new instance (H', y') is still unique. On the other hand, in the highly surjective setting (where $(K/w)^w \gg 2^k$), the number of solutions grows exponentially, and powerful attacks such as

the generalized birthday attack (GBA) can be used to recover a solution.

Relation between SD and RSD depending on k and w. The GV bound and the RSD uniqueness bound allow to clarify the relation between the hardness of the standard syndrome decoding problem (SD) and the regular syndrome decoding problem (RSD). These two bounds delimit three areas depending on the relation between w and k (for a fixed value of K):

- When $2^k \geq \binom{K}{w}$, both the SD and RSD problem are injective with high probability. In this setting, the RSD problem is *easier* than the SD problem. The straightforward reduction works as follows: given an RSD instance (H, y), create a random SD instance (H', y') by shuffling the columns of H and the entries of y, and run the SD solver. By injectivity, the solution x' returned by the solver is the shuffled RSD solution x.
- When $2^k \leq (K/w)^w$, both the SD and RSD problems are surjective with high probability. In this setting, the RSD problem is *harder* than the SD problem. This is because a regular solution to a random SD instance is in particular a solution for the SD problem. More formally, when $2^k \ll (K/w)^w$, one can show that the distribution of random SD instances and RSD instances become both statistically close to uniform. Precisely, a simple calculation shows that their statistical distance to the uniform distribution is at most $\sqrt{\varepsilon}/2$, where $\varepsilon = 2^k/(K/w)^w$ for RSD, and $\varepsilon = 2^k/\binom{K}{w}$ for SD (see *e.g.* [Deb19] for a proof of this statement in the case of SD; the adaptation to RSD is immediate). Therefore, a random SD instance is distributed in particular as a random RSD instance, and running an RSD solver returns a regular solution, which is in particular a valid solution to the SD instance.
- Eventually, when $(K/w)^w \leq 2^k \leq \binom{K}{w}$, the relation between SD and RSD becomes unclear, and their hardness is not directly comparable. In this setting, SD can be attacked efficiently using GBA, while RSD cannot; on the other hand, variants of information set decoding attacks (ISD) can be tuned to benefit from the regularity of the solution.

Furthermore, as previously mentioned, each of SD and RSD is conjectured to be optimally hard on their respective boundaries. The above discussion is summarized in Figure 5.1. The parameter choice is also positioned within this framework: as discussed in Section 5.2.2, w is set to $K/6$, which is optimal in terms of efficiency for the zero-knowledge proof. The value of k is selected to ensure that, with overwhelming probability, the only f-weakly valid solution to RSD is a regular solution, providing security under the standard RSD assumption for the signature scheme. This setting corresponds to being "in the gray zone," not too far from the RSD uniqueness bound.

5.5.2 Known Attacks against RSD

Attention is now directed to existing attacks against the regular syndrome decoding assumption. A detailed overview was provided in [HOS+18], covering three types of attacks: linearization attacks, generalized birthday attacks (GBA), and information set decoding (ISD).

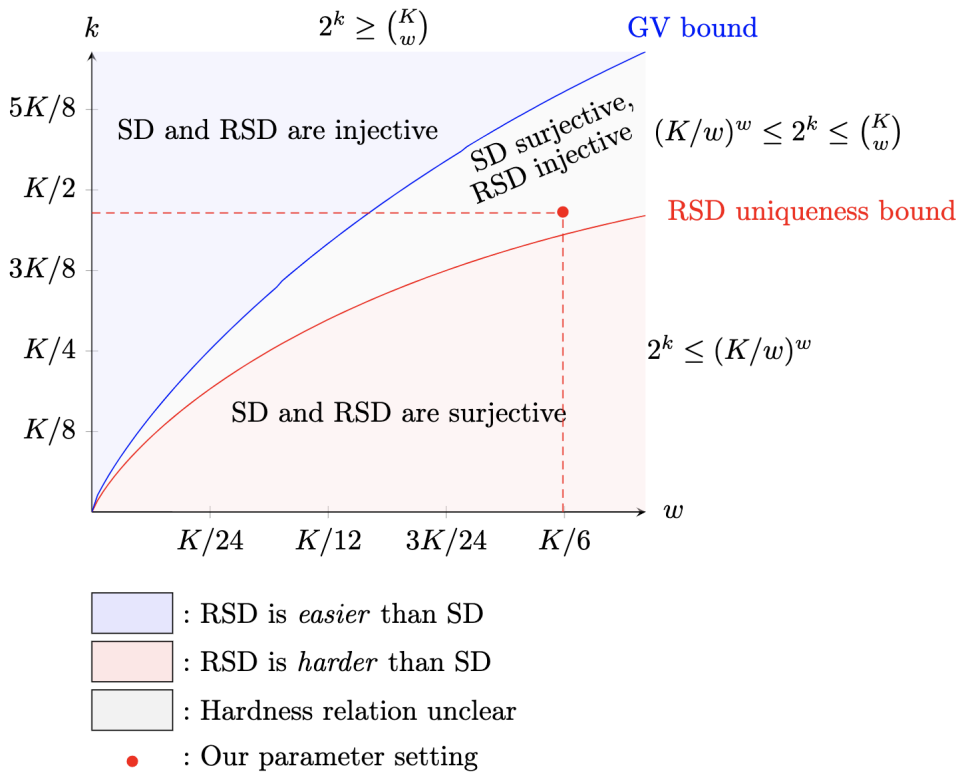

Figure 5.1: Behaviour of SD and RSD when w and k vary, for a fixed value of K

Unfortunately, much of their description appears to conflate the RSD uniqueness bound with the GV bound. This confusion may stem from their consideration of both the SD and RSD problems. Additionally, the attempt to address both SD and RSD attacks simultaneously led to missed opportunities for optimizing the attack by leveraging the regular structure of the noise. In the next section, each of these three attacks is revisited, their exact complexity clarified when applicable in this setting, and tailored specifically to RSD.

5.5.2.1 Generalised Birthday Attack. GBA attacks refer to a family of algorithms based on Wagner's divide-and-conquer algorithm for solving generalized birthday problems. These attacks were first applied to decoding problems by Coron and Joux in [CJ04] and were subsequently improved in [MS09; BLN+09; Kir11; NCB11]. GBA attacks can be directly adapted to RSD. However, they are designed for settings where the problem admits a large number of solutions. Since the focus here is on parameter choices where the average number of solutions is close to 1, GBA attacks do not apply to these parameters, and a full description of these attacks is omitted.

5.5.2.2 Linearization Attack. The linearization attack was introduced by Saarinen in [Saa07]. By leveraging the regular structure of the noise, the attack can be improved by adding a preprocessing phase that reduces the size of the matrix H. The process involves dividing the matrix into blocks of K/w columns of H. The first column of each block is taken and XORed with all other columns, producing a new matrix H'. Additionally, the first column of each block of H is XORed to y, resulting in a new syndrome y'. Let x be any regular solution to $H \cdot x = y$. It can be observed that:

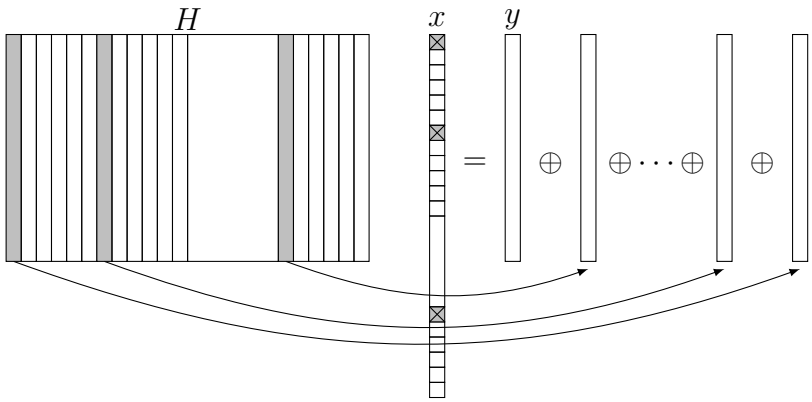

Figure 5.2: Construction of the system $H' \cdot x' = y'$ from the original system $H \cdot x = y$.

- It still holds that $H' \cdot x = y'$, as each block of x has weight 1, ensuring that a single copy of each first column of a block is added to the result when computing $H' \cdot x$.
- The first column of each block of H' is now identically 0 and can therefore be deleted.

After this process, a new matrix $H' \in \mathbb{F}_2^{k \times (K-w)}$ and a new target solution $x' \in \mathbb{F}_2^{K-w}$ are obtained (where x' is derived from x by deleting the first entry of each block, which still uniquely specifies x). The original equation transforms into $H'x' = y'$.
The goal then becomes finding a solution x' such that each block (of size $K/w - 1$) has weight *at most* one. In some cases, deleting certain entries may have eliminated non-zero entries.
In the next step, two new matrices are defined: $H_2 \in \mathbb{F}_2^{k \times (K-w-k)}$, whose columns are randomly selected from the columns of the parity-check matrix H', and $H_1 \in \mathbb{F}_2^{k \times k}$, whose columns are the remaining ones in H'. This leads to $y' = H'x' = H_1x_1 + H_2x_2$, with $x_1 \in \mathbb{F}_2^k$ and $x_2 \in \mathbb{F}_2^{K-w-k}$. At this stage, whenever $x_2 = 0$, the equation simplifies to $y' = H_1x_1$, where H_1 is a square matrix that is invertible with high probability. The solution x_1 is then computed, x' is recovered, and it is verified whether the correct solution has been found. If not, the process is restarted with a new random choice of H_1 and H_2. The complete description of the attack is provided in Protocol 9.

Linearization attacks for RSD

Inputs: A matrix $H \in \mathbb{F}_2^{k \times K}$ a syndrome $y \in \mathbb{F}_2^k$ and a value $w \in \mathbb{N}$.
Output: A regular vector $x \in \mathbb{F}_2^K$ with Hamming weight $\mathsf{HW}(x) = w$ such that $Hx = y$.

1. **Precomputing phase:**
   - In each block of K/w columns of H, delete the first column and XOR it to all remaining columns of the block. Denote the new matrix by H'.
   - XOR all deleted columns to the syndrome y, obtaining a new syndrome y'.

   The goal is now to find a vector x' of length $K - w$ with w blocks, each of Hamming weight 0 or 1, such that $H'x' = y'$.

2. **Permutation phase:**
   - Write $H' = [H_1|H_2]$ and $y' = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ where $H_1 \in \mathbb{F}_2^{k\times k}$.
   - If $H_1^{-1}y'$ is a sparse vector, set $x_1 = H_1^{-1}y'$. Otherwise, shuffle the columns of H' and the entries of y', and start again the permutation phase.

Protocol 9: A variant of linearization attacks tailored to RSD

The expected cost of the attack is computed as follows:

$$\mathbb{E}[\mathsf{cost}] = \frac{\text{cost per iteration}}{\text{success proba per iteration}}.$$

The construction of the matrix H_2 requires selecting $K - w - k$ columns from the total $K - w$ columns of the matrix H'. By selecting the same number of columns for H_2 in each block, $(K - w - k)/w = K/w - 1 - k/w$ columns must be taken from each block. The attack succeeds if all selected columns correspond to zero entries in the solution vector x', meaning that $x_2 = 0$. The probability of this occurrence is now computed.
By construction, each block of x' has a Hamming weight of 0 with probability w/K, and a Hamming weight of 1 otherwise. The expected number of blocks with a Hamming weight of 1 is $(1 - w/K) \cdot w$. The attack succeeds if the $K/w - 1 - k/w$ columns selected in each of these blocks are a subset of the $K/w - 2$ zero-columns. Thus, the probability of selecting only zero entries is approximately given by:

$$P = \left[\frac{\binom{K/w-2}{K/w-1-k/w}}{\binom{K/w-1}{K/w-1-k/w}} \right]^{(1-w/K)\cdot w}.$$

In each iteration, the cost is dominated by the cost of solving a $k \times k$ random system of linear equations, which takes about k^3 arithmetic operations (for realistic values of k, up to a few hundreds), or $k^{2.7}$ (for larger value, using Strassen's algorithm). This yields a total expected cost $\mathbb{E}[\mathsf{cost}] = k^\omega/P$ (with ω an appropriate matrix multiplication exponent).

5.5.2.3 Information Set Decoding Attacks. Information Set Decoding (ISD) is a decoding technique to solve the syndrome decoding problem for linear codes. This type of algorithm was initially introduced in 1962 by Prange [Pra62] and subsequently improved, first through a polynomial improvement proposed by Lee and Leon [LB88; Leo88] and then by an exponential improvement by Stern [Ste88] followed by numerous other improvements [FS09; BLP11; MMT11; BJM+12; MO15].
The structure of ISD attacks is tailored to the specific structure of the target noise vector. As a result, adapting ISD attacks to the RSD setting is not always straightforward. In particular, it is observed that the most efficient and recent variants of ISD [BJM+12; MO15] utilize the representation technique. However, the representation technique is specifically designed for the standard syndrome decoding problem and does not provide any improvement when

applied to *regular* noise. For instance, the representation technique would be effective if different blocks could have, for example, one of two possible weights, as it could introduce additional equations capturing how the weights are distributed. In the context of RSD, however, all blocks have the same Hamming weight of 1, and a direct application of the representation technique does not outperform a standard birthday algorithm. An open question remains, and is left for future work, regarding the possibility of finding an alternative, indirect way of leveraging the representation technique to enhance the attack.

However, it appears that pre-representation technique ISD variants can be adapted to the RSD setting. Furthermore, this adaptation significantly simplifies and improves the attack. At a high level, this is because knowing that the noise is regular significantly reduces the search space. More formally, the adaptation starts from the Generalized ISD algorithm defined by Finiasz and Sendrier [FS09]. This is the same variant that was discussed in [HOS+18]; however, their description appears to confuse the GV bound with the RSD uniqueness bound. Furthermore, a complete description of the attack was not provided, and a few natural optimizations to reduce the search space in the RSD setting were apparently missed (probably because the aim was for a unified description covering both SD and RSD).

Fix an RSD instance (H, y) with noise weight w. The algorithm is a two-step algorithm, with a first step consisting of a Gaussian elimination (which includes a permutation) resulting in a new instance with a new noise vector x', and a second step in which the noise vector is broken into two vectors $x' = x_1||x_2$ (the purpose being to later use the birthday paradox to find a candidate solution x_1). The Finiasz-Sendrier algorithm proceeds by searching for two vectors of the same length as x_1 but with half weight and whose sum is x_2 (using a procedure called *submatrix matching*). In the following, several improvements to this attack are described.

The first observation relates to the choice of partitioning x_1 as a sum of two vectors of the same length, rather than performing the birthday search over vectors whose *concatenation* forms x_1. This choice arises from the random permutation used in the first step: although the initial error x has a regular structure, random permutation removes this regularity. As a result, if x_1 were written as a concatenation of two half-length vectors, there would be no guarantee that each has half of the total Hamming weight. This would introduce an additional overhead, requiring consideration of all possible partitions of the weight w.

This setting avoids this problem. Since the target RSD matrix is provided in systematic form $H = [H'|I_k]$, there is no need to apply a random permutation to H; instead, the initial regular noise vector x can be used directly. This regular structure can be exploited: as the weight is perfectly balanced within each block of the error, x can be rewritten as a concatenation of two vectors $x = x_1||x_2$ with weight $w/2$ each. This significantly reduces the length of the target vector x_1, enabling a birthday paradox search and resulting in a substantial reduction in the total cost of the attack. A full description of the attack is provided in Protocol 10.

Information Set Decoding Algorithm

Inputs: A matrix $H \in \mathbb{F}_2^{k \times K}$ in standard form $H = [H'|I_k]$, a syndrome $y \in \mathbb{F}_2^k$ and a value $w \in \mathbb{N}$.
Output: A vector $x \in \mathbb{F}_2^K$ with Hamming weight $\mathsf{HW}(x) = w$ and such that $Hx = y$.
Parameters: $0 \leq q \leq k$ and $0 \leq p \leq r + q$ with $r = K - k$.
repeat

1. Rewriting the matrix

$$H = \left[H'|I_k\right] = \left[\begin{array}{c|cc} H' & I_q & 0 \\ \hline & 0 & I_{k-q} \end{array}\right] = \left[\begin{array}{cc|c} H_1 & I_q & 0 \\ \hline H_2 & 0 & I_{k-q} \end{array}\right] = \left[\begin{array}{c|c} R_1 & 0 \\ \hline R_2 & I_{k-q} \end{array}\right]$$

   where $R_1 = [H_1|I_q] \in \mathbb{F}_2^{q \times r+q}$ and $R_2 = [H_2|0] \in \mathbb{F}_2^{k-q \times r+q}$ and the vectors

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

   where $y_1 \in \mathbb{F}_2^q, y_2 \in \mathbb{F}_2^{k-q}, x_1 \in \mathbb{F}_2^{r+q}, x_2 \in \mathbb{F}_2^{k-q}$.

2. Computing submatrix problem, solving the reduced problem $R_1 x_1 = y_1$ where $\mathsf{HW}(x_1) = p = \frac{r+q}{K/w}$:
   - Rewrite $R_1 = \bar{R}||\tilde{R}$
   - For all the words $e_1 \in \mathbb{F}_2^{\frac{r+q}{2}}$ of weight $\frac{p}{2} = \frac{r+q}{2K/w}$, create the list $\bar{L} = \{\bar{R}e_1\}$;
   - For all the words $e_2 \in \mathbb{F}_2^{\frac{r+q}{2}}$ of weight $\frac{p}{2} = \frac{r+q}{2K/w}$, create the list $\tilde{L} = \{\tilde{R}e_2 + y_1\}$;
   - Search for a collision between $\bar{L}$ and $\tilde{L}$;
   - For each collision (e_1, e_2) if $\mathsf{HW}(y_2 + R_2(e_1||e_2)) = w - p = \frac{k-q}{K/w}$ then set $x_1 = e_1||e_2$.

3. Extend solution computing $x_2 = y_2 + R_2 x_1$

Protocol 10: An Information Set Decoding algorithm tailored to the RSD setting with matrix in systematic form

Let's now analyze the cost of this attack. The creation of the two lists $\bar{L}$ and $\tilde{L}$ requires, for all the $6^{\frac{r+q}{K/w}}$ possible vectors of length $\frac{r+q}{2}$ and weight $\frac{r+q}{2K/w}$, a matrix-vector multiplication $\bar{R} \cdot e_1, \tilde{R} \cdot e_2$. The total cost for this step is therefore about $(K/w)^{\frac{r+q}{2K/w}} \cdot q \cdot \frac{r+q}{2K/w}$. The search for a collision between the two lists takes linear time. Since both lists are of length $(K/w)^{\frac{r+q}{2K/w}}$, this brings to the total cost a value of $(K/w)^{\frac{r+q}{2K/w}} \cdot q \cdot \frac{r+q}{2K/w}$. The last step in which for each collision the weight is checked by doing a matrix-vector multiplication costs

$\frac{(K/w)^{r+q}}{2^q} \cdot (k-q) \cdot \frac{r+q}{K/w}$. So the total cost of the attack presented in 10 is roughly

$$\left[q \cdot \frac{r+q}{K/w} \cdot (K/w)^{\frac{r+q}{12}}\right] + \left[\frac{(K/w)^{\frac{r+q}{K/w}}}{2^q} \cdot (k-q) \cdot \frac{r+q}{K/w}\right] \tag{5.3}$$

To maximize the possibility of finding a collision between the two lists and at the same time the possibility that the check on the weight in the final step has a positive result, it is necessary that there is no imbalance between the cost of step 2 and the cost of step 3. This can be ensured by requiring that the two parts of 5.3 balance each other. Hence, by imposing $q \cdot \frac{r+q}{K/w} \cdot (K/w)^{\frac{r+q}{2K/w}} = \frac{(K/w)^{\frac{r+q}{K/w}}}{2^q} \cdot (k-q) \cdot \frac{r+q}{K/w}$ the following value is obtained:

$$q\left(1 - \frac{\log_2(K/w)}{2K/w}\right) + \log_2\left(\frac{q}{k-q}\right) = (K-k)\left(\frac{\log_2(K/w)}{2K/w}\right) \tag{5.4}$$

Solving 5.4 for q and injecting the resulting value in 5.3 yields the lowest possible cost for the attack.

5.5.3 An Approximate Birthday Paradox Attack

In this section, a new attack on regular syndrome decoding is described. Informally, the attack resembles the ISD variant discussed in the previous section but applies the birthday paradox algorithm at a different step, removing the need to optimize over the choice of a value q altogether. In exchange, the attack requires more than a standard birthday collision search: it involves an *approximate* collision search between two lists (where approximate means finding two strings whose XOR results in a sparse regular vector — that is, the target strings are identical except for differing in exactly one coordinate per block).
Approximate collision searches have previously been assumed in the literature to take linear time. However, it is important to emphasize that no strict linear time algorithm (in the size of the lists) is currently known for solving this variant of the approximate collision search. Finding such an efficient algorithm is considered an interesting open problem. For the sake of conservativeness in parameter choices, it is *assumed* that an approximate collision can be found in the (linear) time required to scan the lists. It is noted that the existence of a strictly linear time algorithm is far from obvious, and this assumption may therefore be overly conservative.

5.5.3.1 The basic attack. The idea behind the attack is elaborated below. Starting from the RSD instance (H, y) where $H \in \mathbb{F}_2^{k \cdot K}$ and $y \in \mathbb{F}_2^k$ with solution $x \in \mathbb{F}_2^K$ of weight w, H and x are rewritten as follows:

$$H = [H'|I_k] = \left[\bar{H}|\tilde{H}|I_k\right] \text{ and } x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \bar{x} \\ \tilde{x} \\ x_2 \end{bmatrix},$$

where $\bar{x}, \tilde{x} \in \mathbb{F}_2^{\frac{r}{2}}$, and $x_2 \in \mathbb{F}_2^k$. From this point, the attack proceeds in two simple steps:

1. Search step of an "almost collision" between two lists:

$$y = Hx = \left[\bar{H}|\tilde{H}|I_k\right] \begin{bmatrix} \bar{x} \\ \tilde{x} \\ x_2 \end{bmatrix} = \bar{H}\bar{x} + \tilde{H}\tilde{x} + x_2$$

This implies that

$$\left(\tilde{H}\tilde{x} + y\right) + \left(\bar{H}\bar{x}\right) = x_2. \tag{5.5}$$

Since x_2 is a sparse vector, the problem reduces to finding two vectors $\bar{x}, \tilde{x}$ such that $\left(\tilde{H}\tilde{x} + y\right) + \left(\bar{H}\bar{x}\right)$ is sparse. This can be done by storing all possible values of $\left(\tilde{H}\tilde{x} + y\right)$ and $\left(\bar{H}\bar{x}\right)$ in respective lists $\tilde{L}$ and $\bar{L}$, and then searching for an almost-collision between the two lists.

2. Extend the partial solution obtained in the previous step: given $(\bar{x}, \tilde{x})$, set $x_1 = \begin{bmatrix} \bar{x} \\ \tilde{x} \end{bmatrix}$ and compute x_2 as $H'x_1 + y$.

5.5.3.2 Improving the attack by extending the matrix. The basic attack can be improved by allowing the adversary to append additional equations to the parity-check matrix and then bring it back to systematic form (this requires only row operations and does not alter the structure of the solution). At a high level, each additional equation encodes the information that each block of the target noise vector has an odd Hamming weight. Specifically, letting h'_i denote the vector whose entries are all 0, except for a "band of ones" in the i-th block, the inner product between h' and x must be equal to 1. This allows the adversary to add w additional rows to the matrix H before executing the attack described above. The full attack is detailed in Protocol 11.

Approximate Birthday Paradox Attack on RSD

Inputs: A matrix $H \in \mathbb{F}_2^{(k+w)\times K}$ with $H = [H'|M]$, a syndrome $y \in \mathbb{F}_2^{k+w}$, and a value $w \in \mathbb{N}$. It is assumed that the matrix H has been extended with w additional rows and that y has been extended with ones, capturing the fact that each block of x has odd Hamming weight.
Output: A regular vector $x \in \mathbb{F}_2^K$ with Hamming weight $\mathsf{HW}(x) = w$ such that $Hx = y$.
Parameters: $r = K - k - w$

1. Write $H' = \bar{H}|\tilde{H}$;
2. Search for an almost collision:
   - Using all regular words $\bar{x} \in \mathbb{F}_2^{\frac{r}{2}}$ of weight $\frac{r}{2K/w}$, create the list $\bar{L} = \{M^{-1} \cdot \bar{H}\bar{x}\}$;
   - Using all regular words $\tilde{x} \in \mathbb{F}_2^{\frac{r+q}{2}}$ of weight $\frac{r}{2K/w}$, create the list $\tilde{L} = \{M^{-1} \cdot (\tilde{H}\tilde{x} + y)\}$;

- Search for an almost collision $(\bar{x}, \tilde{x})$ between $\bar{L}$ and $\tilde{L}$ and set $x_1 = \bar{x}|\tilde{x}$;

3. Extend the solution by computing $x_2 = y + H'x_1$.

Protocol 11: An Approximate Birthday Paradox Attack on Regular Syndrome Decoding

.

5.5.3.3 Cost of the attack. The second step of the attack, the creation of the lists $\bar{L}$ and $\tilde{L}$ of size $(K/w)^{\frac{r}{2K/w}}$ each, costs $2 \cdot (K/w)^{\frac{r}{2K/w}} \cdot \frac{r}{2K/w} \cdot k = (K/w)^{\frac{r}{2K/w}-1} \cdot r \cdot k$. For the sake of being conservative finding an almost-collision is assumed to take linear time, and bound the cost as

$$\mathsf{cost} \geq (K/w)^{\frac{r}{2K/w}-1} \cdot r \cdot k \tag{5.6}$$

5.5.3.4 A note on optimizing security. In the chosen parameter setting, the above attack significantly outperforms the ISD variant and the linearization attack described in the previous section. Furthermore, the larger the value of k, the more efficient the attack gets. Therefore, the optimal parameter choice is obtained by minimizing the value of k, while maintaining the constraint that the resulting RSD instance does not have too many solutions with high probability (otherwise, GBA attacks could apply). This is in line with the intuition that optimal parameter choices belong to the RSD uniqueness curve (see Figure 5.1).

5.5.4 From RSD to Almost-RSD

The security of the new signature scheme does not directly reduce to the regular syndrome decoding problem but rather to an *almost-regular* syndrome problem. This arises from the fact that the zero-knowledge proof of knowledge described in Section 5.2.2 has a soundness gap: while an honest witness is assumed to be a standard regular vector, the soundness only guarantees that an *f-almost-regular* (or f-almost-antiregular) vector can be extracted from a malicious prover. Here, f-almost-regular means that the extracted vector x is divided into w blocks such that all blocks except f have a Hamming weight of 1, and the f remaining blocks can have any odd weight (in this setting, weights of 1, 3, or 5, since the block size is fixed at 6).
This variant of RSD, informally referred to as *f-almost-RSD*, is a plausible assumption, albeit somewhat exotic. In the main choice of parameters for the signature scheme, the aim is to reduce security to the standard RSD assumption instead. To achieve this, a parameter setting is used where almost-RSD is in fact *equivalent* to RSD. Concretely, the instance (K, k, w) is chosen with two constraints: first, $K/w = 6$ is enforced, and second, k is set as follows:

$$k \leftarrow \left\lceil \log_2 \left(\sum_{i=0}^{f} 6^{w-i} \cdot \binom{w}{i} \cdot 26^i \right) \right\rceil + \lambda.$$

This guarantees that $\sum_{i=0}^{f} 6^{w-i} \cdot \binom{w}{i} \cdot 26^i \leq 2^{k-\lambda}$. Intuitively, without the $+\lambda$ term, the above would correspond to the *f-almost-RSD uniqueness bound*, *i.e.* the threshold above which there is, on average, a single almost-regular solution to a random RSD instance. The

left-hand side of the inequality represents the total number of regular vectors with $w - f$ blocks of weight 1 and f blocks of weight 1, 3, or 5 — that is, the space of solutions with the correct structure. Let (H, y) be a random RSD instance, where y is computed as $H \cdot x$ for a random regular vector x. With the $+\lambda$ term, the inequality guarantees that the probability of any additional solution beyond x existing is at most $2^{-\lambda}$. In this regime, the almost-RSD problem becomes equivalent to the standard RSD problem: except with probability $2^{-\lambda}$, any almost-regular solution to the problem must be the only regular solution.

Chapter 6

Enhancements via PPRF and Efficient MPC Protocols

This chapter presents two contributions to the field of signature schemes constructed from the MPC-in-the-head (MPCitH) paradigm:

- The notion of multi-instance puncturable pseudorandom function (PPRF) is introduced, along with an efficient instantiation from the AES block cipher, formally proven secure in the ideal cipher model. This construction can replace the hash-based PPRF used in most previous MPCitH signatures, improving both signing time and verification time (e.g., from $12\times$ to $55\times$ in experiments with the recent scheme of [HJ24]).
- The MPCitH signature based on the regular syndrome decoding (RSD) problem, introduced in the previous chapter, is enhanced in all aspects by incorporating the new PPRF structure. This improvement results in its unforgeability being tightly reduced to the multi-instance security of the underlying PPRF, showcasing how this new primitive leads to better security reduction.

Contents

6.1 Problem Statement

All state-of-the-art MPCitH signature schemes rely on a puncturable pseudorandom function (PPRF), which allows generating a large number n of pseudorandom strings such that, given an index i, the signer can reveal all pseudorandom values except the i-th one using an "opening" of size $\lambda \cdot \log n$ (where λ is a security parameter). The *de facto* PPRF originally used in prominent works such as Picnic [ZCD+20] was the GGM PPRF [GGM86], where the PRF evaluation on input i with key K is the i-th leaf of a full binary tree with root labeled with K. In GGM, the labels of the two children of a node x are computed as $\mathsf{F}(x) = (\mathsf{F}_0(x), \mathsf{F}_1(x))$ using a length-doubling pseudorandom generator (PRG) $x \mapsto (\mathsf{F}_0(x), \mathsf{F}_1(x))$. The PRG is typically instantiated as $x \mapsto (x \oplus \pi_0(x), x \oplus \pi_1(x))$ for a pair of random invertible permutations (π_0, π_1): this instantiation yields a provably secure construction in the random permutation model [GKW+20] and enables fast expansion when instantiating the permutations using the AES block cipher with a fixed key (leveraging hardware acceleration). However, as observed in [DN19], in the context of signatures, this choice allows for a devastating multi-target attack: after 2^t signature queries, an attacker can find the root of one of the GGM trees using 2^{128-t} work on average, and recover the secret signing key upon finding a collision.

In response to this vulnerability, Picnic [ZCD+20] and most recent MPCitH signature schemes, including BBQ [DDO+19], Banquet [BDK+21], and the NIST post-quantum candidates (e.g., SDitH [CHT23], MIRA [ABC+23], MiRitH [ABB+23], MQOM [BFR23], PERK [GSR23], Ryde [DGO+23], and Biscuit [BKP+23]), switched to hash-based PRGs, e.g., $\mathsf{F}_b(x, i, \mathsf{salt}) \leftarrow H(x \| i \| b \| \mathsf{salt})$, where i is the parent node index and salt is a random salt included in the signature. Unfortunately, replacing AES with a hash function incurs significant performance overhead, being up to $50\times$ slower according to [GKW+20].

The following simple observation forms the foundation for this chapter's contributions: in the AES-based instantiation, instead of relying on a global fixed key for all instances (as in Picnic [ZCD+20]), a per-signature random AES key can be used as the salt. This eliminates costly re-keying at every tree node while maintaining block size efficiency. While the idea of rotating the key is not particularly novel (it has been mentioned in contexts such as [Roy22]), it motivates the introduction of multi-instance PPRFs, defined as $\mathsf{F}(x, K_0, K_1) = (x \oplus \mathsf{AES}_{K_0}(x), x \oplus \mathsf{AES}_{K_1}(x))$, to capture the security requirements needed to prevent multi-target attacks precisely. Unlike previous works that directly proved full signature constructions in the ROM, this approach adopts a more modular methodology. Furthermore, a formal proof establishes the multi-instance security of the AES-based PPRF in the ideal cipher model using Patarin's H-coefficient technique [Pat09; CS14].

This analysis induces a security loss of $\log D + 3$ bits, where D is the GGM tree depth. For example, if $D = 8$, the loss is 6 bits. The 3-bit loss stems from bounding the attacker's worst-case runtime ($2^{-\lambda}$ probability implies 2^{λ} runtime), reducible to 1 bit for expected runtime. The $\log_2 D$ loss arises from the PRG-to-PRF reduction and is inherent to the construction. To mitigate the $\log_2 D$ loss, a variant is proposed: D key pairs $(K_0^i, K_1^i)_{i \le D}$, derived from a PRG-stretched 2λ-bit salt, are assigned to each tree level. This increases AES rekeyings from 2 to $2D$, though the cost is negligible (with $D \in [8, 16]$). It is conjectured that this variant achieves $\lambda - 3$ bits of security ($\lambda - 1$ for expected runtime), though a direct proof in

the ideal cipher model remains an open problem.
Replacing hash-based PPRFs with the AES-based construction is expected to yield substantial efficiency gains, especially where GGM tree expansion dominates runtime. Testing on the scheme of [HJ24], presented at CRYPTO'24, shows runtime improvements of $12\times$ for $D = 8$ and $55\times$ for $D = 16$, as shown in Table 6.1.

[HJ24]	D	τ	$\lvert\sigma\rvert$	signing	verification
hash-based PPRF	8	16	6.2 kB	9.24 ms	9.11 ms
	16	8	4.1 kB	1.1 s	1.1 s
AES-based PPRF	8	16	6.2 kB	0.80 ms	0.71 ms
(this work)	16	8	4.1 kB	19.5 ms	19.2 ms

Table 6.1: Case analysis of the impact of using the faster AES-based multi-instance PPRF on the signature scheme of [HJ24] for two sets of parameters: $D = 8$ (fast signing) and $D = 16$ (short signatures) compared to the standard hash-based construction. D denotes the depth of the GGM tree (equivalently, 2^D corresponds to the number of virtual parties in the MPC protocol run "in the head"), and τ to the number of repetitions to achieve 128 bits of security. All schemes run on one core of an AMD EPYC 9374F processor clocked at 3.85GHz.

In this chapter the signature presented in the previous chapter is significantly improved in several aspects, using several new techniques to provide more flexible parameter choices and improved performances. Of course, the first ingredient is the just improved (PPRF) tailored to MPC-in-the-head signatures that uses a salted-GGM tree based on AES, which achieves the same performances as the best *unsalted* GGM PPRF constructions, without suffering from collision-based attacks.
To obtain improved performances compared to the previous signature, the regular syndrome decoding instances are encoded using a sparse representation on top of the dense representation used in the older one. Sparsely encoding regular syndrome decoding instances is quite natural and relies on the use of an indicator vector to locate the non-zero positions. However, such a representation is not compatible with the secret sharing techniques that are used to split the key between the virtual parties that are introduced by the MPC-in-the-head paradigm: to use sparse representations, it is necessary to develop new conversion techniques involving both types of representations. Along the way, the presented scheme relies on a mechanism to prevent cheating behavior in the conversion, which requires a highly non-trivial combinatorial analysis. Overall, the new signature scheme is more than 30% shorter compared to the scheme of Protocol 6 and can use significantly more conservative parameter sets, for similar runtimes.
The implementation of the signature scheme is a proof-of-concept and does not use any optimizations such as batching, vectorization, or bit slicing. Nevertheless, it confirms that the scheme exhibits excellent performance. Since for the previous signature (Section 5.4) there is no implementation, the scheme is compared to SDitH, the state-of-the-art MPCitH signature from syndrome decoding [CHT23], that makes use of batching techniques and advanced hardware instructions. In addition, tight estimates are provided for the performance improvements resulting from integrating the fast folding optimization of [HJ24] into this scheme and into SDitH (while the implementation of [HJ24] is used to obtain runtimes for the faster folding, it is noted that a full-fledged implementation of the scheme

integrating the folding optimization is not yet available). Below is a sample of parameters:

- (fast) signature size 6.5kB, signing time 1.40 ms
- (medium) signature size 5.7 kB, signing time 3.56 ms
- (compact) signature size 4.9 kB, signing time 23.9 ms.

Refer to Table 6.4 for more details on parameters and implementation.
The scheme is compared to SDitH [AGH+23], the fastest known code-based signature scheme to date, by running both schemes on the same hardware and for comparable parameter sets. To better isolate the effect of the improved PPRF, SDitH is also benchmarked with their PPRF replaced by the improved construction,[1] as well as this scheme using the hash-based PPRF of SDitH. For both, the folding optimization of [HJ24] is integrated. Benchmarks are summarized in Table 6.2. Even when comparing the unoptimized implementation to the optimized implementation of [CHT23], $3\times$ to $4\times$ runtime improvements for $D = 8$ (with signatures of comparable size) are observed.

	D	τ	$\lvert\sigma\rvert$	signing time
SDitH (hash-PPRF)	8	17	8.2 kB	6.82 ms
	12	11	6.0 kB	46.8 ms
SDitH (AES-PPRF)	8	17	8.2 kB	6.05 ms
	12	11	6.0 kB	37.9 ms
SDitH (AES-PPRF+)	8	17	8.2 kB	6.03 ms
	12	11	6.0 kB	31.5 ms
This scheme (hash-PPRF)	8	17	7.8 kB	4.07 ms
	12	11	6.1 kB	43.83 ms
This scheme (AES-PPRF)	8	17	7.8 kB	1.65 ms
	12	11	6.1 kB	19.1 ms
This scheme (AES-PPRF+)	8	17	7.8 kB	0.64 ms
	12	11	6.1 kB	2.13 ms

Table 6.2: Comparison of the new signature scheme with SDitH for $D = 8$ and $D = 12$, with and without the improved multi-instance puncturable pseudorandom function (denoted AES-PPRF and hash-PPRF respectively) and with or without integrating the folding optimization of [HJ24] (denoted AES-PPRF+ and AES-PPRF respectively). All schemes were run on one core of an Intel Core i7 processor 14700KF at 3.4 GHz frequency.*

[1] In the conference version of their work, the construction of [AGH+23] initially used an *unsalted* GGM tree (instantiated using AES), which is shown to be insecure (with a concrete attack that breaks the scheme using 2^{40} signatures in time 2^{69}). The authors later fixed this issue in their NIST submission [CHT23], using a proper salted GGM tree instantiated with a hash function.

6.2 New Puncturable Pseudorandom Functions for MPCitH

The starting point is the GGM puncturable pseudorandom function [KPT+13; BW13; BGI14; GGM86], which is used in all modern MPC-in-the-head protocol, in which the prover generates shares of the witness –eventually together with shares of some appropriate preprocessing material– to be distributed among the n virtual parties such that, in the last round the prover will reveal $n-1$ out of n shares to the verifier.
Since a *puncturable pseudorandom function* (PPRF) is a PRF F such that given an input x, and a PRF key k, allows evaluating F at every point except for x, (see section 3.4.5 for more detail) it is clear that these functions can be used to significantly optimize the last step of MPCitH protocols: using a PPRF, the prover can define all seeds sd_i as outputs of the PRF, using a master seed sd^* as the PRF key. Then, revealing the key sd^* punctured at a point i suffices to succinctly reveal all seeds $(\mathsf{sd}_j)_{j\neq i}$ while hiding sd_i. Concretely, using the GGM PPRF [GGM86], the prover generates N seeds $\mathsf{sd}_1, \cdots, \mathsf{sd}_N$ as the leaves of a binary tree of depth $\lceil \log_2 N \rceil$, where the two children of each node are computed using length-doubling pseudorandom generators. This way, revealing all seeds except sd_i requires only sending the seeds on the nodes along the co-path from the root to the i-th leave, which reduces the communication from $\lambda \cdot (N-1)$ to $\lambda \cdot \lceil \log_2 N \rceil$.

Salted GGM Tree Unfortunately, MPC-in-the-head can suffer from collision attacks if the GGM PPRF is used *as is*, as shown in [DN19]: after approximately $2^{\lambda/2}$ signature queries, collisions in the master seed sd^* may leak the secret signing key. To address this, previous works have introduced a 2λ-bit salt, but its integration into the GGM PPRF is inconsistent, leading to either ambiguity or insecurity. Specifically:

- In Banquet [BDK+21], and in the more recent work of [AGH+23], the seeds $(\mathsf{sd}_1, \cdots, \mathsf{sd}_n)$ are generated from an *unsalted* GGM PPRF, and the salt is only used at the leaves, when stretching the share of each party P_i from its seed as $\mathsf{PRG}(\mathsf{sd}_i, \mathsf{salt})$.
- In [FJR22] and in 5.4, the signature description loosely states that $(\mathsf{sd}_1, \cdots, \mathsf{sd}_n)$ are generated in a tree-based fashion using the master seed sd^* and the salt salt. However, the way the salt is used within the GGM construction is not specified precisely.

It is observed that using the salt only at the leaves, as in [BDK+21; AGH+23], does not shield the signature from collision attacks. The attack proceeds as follows:

- An attacker queries m signatures. Each signature will contain some number τ of $\lceil \log_2 N \rceil$-tuples of intermediate PRG evaluations (corresponding to the seeds on the co-path to the unopened leave; τ corresponds to the number of repetitions of the underlying identification scheme). Let $(\mathsf{sd}^1, \cdots, \mathsf{sd}^k)$ denote all seeds received this way, where $k = m \cdot \tau \cdot \lceil \log_2 N \rceil$.
- The attacker locally samples random seeds sd and evaluates its two children $(\mathsf{sd}_0, \mathsf{sd}_1) \leftarrow\!\$\ \mathsf{PRG}(\mathsf{sd})$, until one of the sd_b collides with some sd^i.

- Since the preimage of sd_b is known, the parent seed of sd^i is recovered, from which the seed associated with the unopened leave in one of the signatures can be computed.
- Given this seed, and using the salt salt associated with the signature (which is public), the attacker reconstructs all virtual parties' shares and reconstructs the secret witness (the AES secret key in [BDK+21], the syndrome decoding solution in [AGH+23]). Using the witness, arbitrary signatures can now be forged.

As is clear from the above description, adding salt to the leaves has absolutely *no* effect on the security of the signature against this collision attack. Efficiency-wise, after receiving m signatures, an attacker finds a collision in time $2^\lambda/(m \cdot \tau \cdot \lceil \log_2 N \rceil)$. For example, using $\lambda = 128$, $n = 2^{16}$, and $\tau = 9$ (this is a parameter set from [AGH+23]), after seeing only $m = 2^{40}$ signatures, the scheme can be broken in time $\approx 2^{69}$.
The attack described appears to stem from presentation issues in the respective papers, as some implementations already address it. For instance, Banquet[2] includes salt within all intermediate tree computations, and the updated NIST submission [CHT23] based on [AGH+23] resolves the problem in its implementation.
However, efficiency concerns remain: unsalted GGM trees can use fixed-key AES efficiently with Intel AES-NI [GKW+20], but adding salt complicates its usage. Instead, implementations such as Picnic [ZCD+20], BBQ [DDO+19], Banquet [BDK+21], and recent schemes based on [AGH+23] rely on a hash function (such as SHAKE): $\mathsf{sd}_b \leftarrow H(\mathsf{sd}|i|j|b|\mathsf{salt})$, where i is the parent node index, and $j \leq \tau$ is a counter for the identification scheme repetitions. This approach, however, is up to 50× slower than using AES (see Table 6.3), which significantly impacts protocols like those in [AGH+23], where tree generation dominates signing time.

D	τ	AES	SHA3	SHAKE
9	16	0,04ms	26,59ms	39,42ms
11	13	0,11ms	66,1ms	100ms
13	11	0,21ms	122ms	179,77ms
15	10	0,21ms	122,95ms	178,2ms

Table 6.3: Comparison of the utilization speeds of AES and hash functions in the instantiation of the GGM tree.

A fast salted GGM tree in the ideal cipher model This chapter introduces a new salted GGM tree construction that matches the efficiency of the fastest *unsalted* GGM trees while offering stronger security guarantees. A formal security proof (Theorem 6.2.1) demonstrates that the construction is a *multi-instance secure* PPRF, a notion introduced in section 6.2.1. Unlike standard PPRFs, which incur a security loss proportional to the number of signature queries (as illustrated by the attack), the unforgeability of MPCitH signatures reduces tightly to the multi-instance security of the PPRF.
The multi-instance PPRF is based on a very simple idea: using the top-performing GGM construction from a fixed-key block cipher and treating the cipher key as the salt. While

[2] https://github.com/dkales/banquet

intuitive, the security proof (Theorem 6.2.1), conducted in the ideal cipher model, is technically challenging. It relies on the H-coefficient technique of Patarin [Pat09; CS14] and combines it with a balls-and-bins analysis to measure the number of seed and cipher key collisions and tightly estimate their impact on security.

A PPRF in the Random Permutation Model

The starting point is a PPRF construction from [GKW+20], which is a tweak on the original GGM construction, where the PRG is instantiated with the following "Davies-Meyer" function:

$$G : x \to (\pi_0(x) \oplus x, \pi_1(x) \oplus x)\,.$$

In this construction, (π_0, π_1) are two fixed pseudorandom permutations. Using this PRG, the construction of the PPRF proceeds in a tree-based fashion: a PPRF key $K \leftarrow\!\!\$\ \{0,1\}^\lambda$ is sampled. On input $x = (x_1, \cdots, x_n)$, the PPRF F_K returns

$$G_{x_n}(G_{x_{n-1}}(\cdots G_{x_1}(K)\cdots)),$$

where G_0, G_1 denote the left and right half of the output of G, respectively. Puncturing x is done by computing all values on the co-path to x in the tree, *i.e.*, the values

$$G_{\bar{x_i}}(G_{x_{i-1}}(\cdots G_{x_1}(K)\cdots))$$

for $i = 2$ to n: knowing the values on the co-path allows reconstructing the entire tree except for $F_K(x)$, whose values are pseudorandom under the security of G. To prove the security of the construction, the authors of [GKW+20] rely on the *random permutation model*, where (π_0, π_1) are modeled as two independent random permutations.
In [GKW+20], the motivation for introducing the construction is that, in practice, π_0, π_1 can be instantiated using the AES block cipher with two fixed keys (K_0, K_1). This allows G to be evaluated using two calls to AES, which is extremely fast when using the AES-NI hardware instruction set (encrypting with AES using AES-NI takes as little as *1.3 cycles per byte* according to [MSY21]). Furthermore, the entire construction requires only two executions of the AES key schedule. This GGM construction remains, by a significant margin, the fastest known PPRF and has been featured extensively in recent works on function secret sharing [GI14; BGI15; BGI16b; BGI19; BCG+21], pseudorandom correlation generators [BCG+18; BCG+19b; BCG+19a; BCG+20b; YWL+20; WYK+21; CRR21; BCG+22], and many others. It is also the construction suggested in [AGH+23], though, as shown above, it is insecure in the context of signatures.
Observing that this fast PPRF construction is typically instantiated using a block cipher suggests the following idea, which is very natural in retrospect: use the above construction but instantiate (π_0, π_1) using a block cipher (such as AES) and *use the block cipher keys* (K_0, K_1) *as a random salt.* This means that in each instance, the pair (K_0, K_1) will be sampled at random. When using AES, this introduces no changes to the efficiency of the construction, since, in each instance, the AES key schedule still needs to be executed only twice. Yet, with this modification, there is potential for the use of fresh cipher keys in distinct instances to prevent collision attacks.

6.2.1 Multi-istance PPRFs and PRGs

To formalize this idea, in the following sections, the notion of *multi-instance* puncturable pseudorandom function is introduced. An efficient construction from a block cipher is described, and its security is formally proven in the ideal cipher model.

Multi-instance PPRFs

At a high level, an N-instance PPRF is a PPRF that additionally takes as input a random salt: in the N-instance security game N keys $(k_1, \cdots, k_N)$, inputs $(x_1, \cdots, x_N)$, and salts $(\mathsf{salt}_1, \cdots, \mathsf{salt}_N)$ are sampled randomly, along with a bit $b \leftarrow\$ \{0,1\}$. The adversary is given $((x_1, \mathsf{salt}_1), \cdots, (x_N, \mathsf{salt}_N))$ and the punctured keys $(k_1\{x_1\}, \cdots, k_N\{x_N\})$. If $b = 0$, it additionally receives $(F_K(x_1, \mathsf{salt}_1), \cdots, F_K(x_N, \mathsf{salt}_N))$; otherwise, it receives N random outputs $(y_1, \cdots, y_N)$. The adversary wins by correctly guessing b. The PPRF is N-instance (t, ε)-secure if no t-time adversary has an advantage greater than ε.
For constructions involving τ parallel calls to the PPRF with the same salt –such as signatures constructions that involve τ parallel instances of the PPRF with the same salt– the notion extends to (N, τ)-instance security, accounting for N instances of τ repetitions, where the salt varies across instances –signature queries– but remains constant within repetitions.
The formal definition is below.

Definition 6.2.1 ((N, τ)-instance (t, ϵ)-secure PPRF)**.** *A function family $F = \{F_K\}$ with input domain $[2^D]$, salt domain $\{0,1\}^s$, and output domain $\{0,1\}^\lambda$, is an (N, τ)-instance (t, ϵ)-secure PPRF if it is a PPRF which additionally takes as input a salt* salt*, and for every non-uniform PPT distinguisher $\mathcal{D}$ running in time at most t, it holds that for all sufficiently large λ,*

$$\mathsf{Adv}^{\mathsf{PPRF}}(\mathcal{D}) = |\Pr[\mathsf{Exp}^{\mathsf{rw\text{-}pprf}}_{\mathcal{D}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{iw\text{-}pprf}}_{\mathcal{D}}(\lambda) = 1]| \leq \epsilon(\lambda)$$

where the experiments $\mathsf{Exp}^{\mathsf{rw\text{-}pprf}}_{\mathcal{D}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{iw\text{-}pprf}}_{\mathcal{D}}(\lambda)$ are defined below.

$\underline{\mathsf{Exp}^{\mathsf{rw\text{-}pprf}}_{\mathcal{D}}(\lambda)}$:	$\underline{\mathsf{Exp}^{\mathsf{iw\text{-}pprf}}_{\mathcal{D}}(\lambda)}$:
• $((K_{j,e})_{j\leq N, e\leq\tau} \leftarrow\$ (\{0,1\}^\lambda)^{N\cdot\tau}$	• $((K_{j,e})_{j\leq N, e\leq\tau} \leftarrow\$ (\{0,1\}^\lambda)^{N\cdot\tau}$
• $\mathsf{salt} := (\mathsf{salt}_1, \ldots, \mathsf{salt}_N) \leftarrow\$ \{0,1\}^s$	• $\mathsf{salt} := (\mathsf{salt}_1, \ldots, \mathsf{salt}_N) \leftarrow\$ \{0,1\}^s$
• $\mathbf{i} := ((i_{1,e})_{e\leq\tau}, \ldots, (i_{N,e})_{e\leq\tau}) \leftarrow\$ [2^D]^{N\cdot\tau}$	• $\mathbf{i} := ((i_{1,e})_{e\leq\tau}, \ldots, (i_{1,e})_{e\leq\tau}) \leftarrow\$ [2^D]^{N\cdot\tau}$
• $\forall j \leq N, e \leq \tau : K^{i_{j,e}}_{j,e} \leftarrow F.\mathsf{Punc}(K_{j,e}, i_{j,e})$	• $\forall j \leq N, e \leq \tau : K^{i_{j,e}}_{j,e} \leftarrow F.\mathsf{Punc}(K_{j,e}, i_{j,e})$
• $(y_{j,e})_{j\leq N, e\leq\tau} \leftarrow (F_{K_{j,e}}(i_{j,e}, \mathsf{salt}_j))_{j\leq N, e\leq\tau}$	• $(y_{j,e})_{j\leq N, e\leq\tau} \leftarrow\$ (\{0,1\}^\lambda)^{N\cdot\tau}$
Output $b \leftarrow \mathcal{D}(\mathsf{salt}, \mathbf{i}, (K^{i_{j,e}}_{j,e}, y_{j,e})_{j\leq N, e\leq\tau})$	**Output** $b \leftarrow \mathcal{D}(\mathsf{salt}, \mathbf{i}, (K^{i_{j,e}}_{j,e}, y_{j,e})_{j\leq N, e\leq\tau})$

The actual construction satisfies a stronger property, wherein indistinguishability is preserved even when the ideal world experiment not only samples $(y_1, \cdots, y_N)$ uniformly at random but also samples "fake" punctured keys $K^{x_k}_j$ uniformly at random over an appropriate domain. While this stronger notion is not strictly necessary for the signature

construction, its use simplifies the analysis. The definition is explicitly stated below for the punctured key domain corresponding to the (GGM-based) construction, but the notion extends naturally to arbitrary domains.

Definition 6.2.2 ((N,τ)-instance strongly (t,ϵ)-secure PPRF)**.** *A function family $F = \{F_K\}$ with input domain $[2^D]$, salt domain $\{0,1\}^s$, output domain $\{0,1\}^\lambda$, and punctured key domain $(\{0,1\}^\lambda)^D$ is an (N,τ)-instance (t,ϵ)-secure PPRF if it is a PPRF which additionally takes as input a salt* salt*, and for every non-uniform PPT distinguisher $\mathcal{D}$ running in time at most t, it holds that for all sufficiently large λ,*

$$\mathsf{Adv}^{\mathsf{PPRF}}(\mathcal{D}) = |\Pr[\mathsf{Exp}^{\mathsf{rw\text{-}pprf}}_{\mathcal{D}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{iw\text{-}spprf}}_{\mathcal{D}}(\lambda) = 1]| \leq \epsilon(\lambda),$$

where the experiment $\mathsf{Exp}^{\mathsf{iw\text{-}spprf}}_{\mathcal{D}}(\lambda)$ is defined as $\mathsf{Exp}^{\mathsf{iw\text{-}pprf}}_{\mathcal{D}}(\lambda)$, except that the line $\forall j \leq N, e \leq \tau : K^{i_{j,e}}_{j,e} \leftarrow F.\mathsf{Punc}(K_{j,e}, i_{j,e})$ is replaced by $\forall j \leq N, e \leq \tau : K^{i_{j,e}}_{j,e} \leftarrow\!\!\$\ (\{0,1\}^\lambda)^D$.

As a first step toward proving the security of the construction, the similar (but simpler) notion of (N,τ)-instance (t,ε)-secure PRG is introduced.

Multi-Instance PRGs In this section, the notion of (N,τ)-instance (t,ε)-secure pseudorandom generator is introduced, extending the concept of pseudorandom generators to the multi-instance setting (with salt) analogously to the definition of multi-instance PPRFs. It is then demonstrated that the standard GGM construction extends immediately to the multi-instance setting: (length-doubling) (N,τ)-instance (t,ε)-secure PRGs imply (N,τ)-instance strongly $(t, D\cdot\varepsilon)$-secure PPRFs with input domain $[2^D]$ and punctured key domain $(\{0,1\}^\lambda)^D$. The process begins by defining (N,τ)-instance (t,ϵ)-secure length-doubling PRGs. To interface more easily with the tree-based GGM construction of PPRFs, $(\mathsf{F}_0,\mathsf{F}_1)$ is used to denote functions that compute the left half and right half of the length-doubling PRG output.

Definition 6.2.3 ((N,τ)-instance (t,ϵ)-secure PRG)**.** *A PRG $\mathsf{PRG} = (\mathsf{F}_0,\mathsf{F}_1)$ with $\mathsf{F}_b : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ is an (N,τ)-instance (t,ϵ)-secure length-doubling* PRG *if for every non-uniform PPT distinguisher $\mathcal{D}$ running in time at most t, it holds that for all sufficiently large λ,*

$$\mathsf{Adv}^{\mathsf{PRG}}(\mathcal{D}) = |\Pr[\mathsf{Exp}^{\mathsf{rw\text{-}prg}}_{\mathcal{D}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{iw\text{-}prg}}_{\mathcal{D}}(\lambda) = 1]| \leq \epsilon(\lambda),$$

where $\mathsf{Exp}^{\mathsf{rw\text{-}prg}}_{\mathcal{D}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{iw\text{-}prg}}_{\mathcal{D}}(\lambda)$ are defined below.

$\underline{\mathsf{Exp}^{\text{rw-prg}}_{\mathcal{D}}(\lambda)}$:	$\underline{\mathsf{Exp}^{\text{iw-prg}}_{\mathcal{D}}(\lambda)}$:
• $(\mathsf{salt}_1, \mathsf{salt}_2, \ldots, \mathsf{salt}_{2N}) \leftarrow_r \{0,1\}^\lambda$	• $(\mathsf{salt}_1, \mathsf{salt}_2, \ldots, \mathsf{salt}_{2N}) \leftarrow_r \{0,1\}^\lambda$
• $(\mathsf{sd}_{i,e})_{i\le N, e\le\tau} \leftarrow_r (\{0,1\}^\lambda)^{N\cdot\tau}$	
• $\forall i \le N, e \le \tau$:	• $(y_{i,e})_{i\le 2N, e\le\tau} \leftarrow_r (\{0,1\}^\lambda)^{2N\cdot\tau}$
– $y_{2i-1,e} \leftarrow \mathsf{F}_0(\mathsf{sd}_{i,e}, \mathsf{salt}_{2i-1})$	
– $y_{2i,e} \leftarrow \mathsf{F}_1(\mathsf{sd}_{i,e}, \mathsf{salt}_{2i})$	
Output $b \leftarrow \mathcal{D}\big((\mathsf{salt}_i, (y_{i,e})_{e\le\tau})_{i\le 2N}\big)$	**Output** $b \leftarrow \mathcal{D}\big((\mathsf{salt}_i, (y_{i,e})_{e\le\tau})_{i\le 2N}\big)$

The definition extends naturally to PRGs that stretch their seeds by a larger factor. Additionally, in the definition above, it is assumed that each of F_0 and F_1 takes a distinct λ-bit salt. While the definition can be generalized to accommodate more flexible salting procedures, it is formulated concerning the specific use of salt in the actual construction for notational simplicity. Notably, the fact that each F_b operates with only λ bits of salt is a crucial aspect resulting from the use of block ciphers, which also makes the security analysis significantly more complex.

6.2.2 Contrusction of a multi-instance PPRF using a multi-instance PRG

Given a seed $\mathsf{sd} \leftarrow\!\!\$\ \{0,1\}^\lambda$, salt $\mathsf{salt} := (\mathsf{salt}_0, \mathsf{salt}_1) \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$, and a multi-instance secure PRG $\mathsf{F}_0, \mathsf{F}_1 : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$, a PPRF $\mathsf{PPRF}(\mathsf{sd}, \mathsf{salt}) = \mathsf{PPRF}(\mathsf{sd}, \mathsf{salt}, 2^D)$ over input domain $\{0,1\}^D$ (later identified with $[2^D]$) is recursively defined in a tree-based manner as follows:

- The first layer includes two nodes $X_0 := \mathsf{F}_0(\mathsf{sd}, \mathsf{salt}_0),\ X_1 := \mathsf{F}_1(\mathsf{sd}, \mathsf{salt}_1)$.
- Each layer of the tree is constructed from the nodes of the previous layer similarly, as follows:

$$\begin{aligned}\mathsf{PPRF}_{\mathsf{sd}}(\mathsf{salt}, i) &= \mathsf{F}_{i_D}\left(\mathsf{PPRF}_{\mathsf{sd}}\left(\mathsf{salt}, i_1, \ldots, i_{D-1}\right), \mathsf{salt}\right) \\ &= \mathsf{F}_{i_D}\left(\mathsf{F}_{i_{D-1}}\left(\ldots\left(\mathsf{F}_{i_1}(\mathsf{sd}, \mathsf{salt}), \mathsf{salt}\right), \mathsf{salt}\right), \mathsf{salt}\right),\end{aligned}$$

  where $i_1, \cdots, i_D$ denote the bits of i.

As in the standard GGM construction, a punctured key at i is the co-path to i in the tree, *i.e.*, the set of intermediate nodes that can be used to recover all leaves except the i-th one:$\mathsf{CoPath}_{\mathsf{sd}}(\mathsf{salt}, i) = \mathsf{PPRF}_{\mathsf{sd}}\left(\mathsf{salt}, i_{1,\ldots,\bar{j}}\right)_{j=1,\ldots,D}$. The formal construction is presented in Protocol 12.
As in the standard GGM construction, a punctured key at i corresponds to the co-path to i in the tree, *i.e.*, the set of intermediate nodes that allow the recovery of all leaves except the i-th one: $\mathsf{CoPath}_{\mathsf{sd}}(\mathsf{salt}, i) = \mathsf{PPRF}_{\mathsf{sd}}\left(\mathsf{salt}, i_{1,\ldots,\bar{j}}\right)_{j=1,\ldots,D}$. The formal construction is

presented in Protocol 12, and the proof of security is provided in Theorem 6.2.1. The proof is a natural extension of the security analysis of the GGM construction [GGM86].

New PPRF

Parameters:

- Two functions $\mathsf{F}_0, \mathsf{F}_1 : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$. Number of leaves $n = 2^D \in \mathbb{N}$.

Construction:

- Sample $(\mathsf{sd}, \mathsf{salt}) \leftarrow\!\!\$\ \{0,1\}^{3\lambda}$ where $\mathsf{salt} := (\mathsf{salt}_0, \mathsf{salt}_1)$ where $\mathsf{salt}_0, \mathsf{salt}_1$ are used for $\mathsf{F}_0, \mathsf{F}_1$ respectively. For simplicity, $\mathsf{F}_i(\mathsf{sd}, \mathsf{salt}_i)$ is written as $\mathsf{F}_i(\mathsf{sd}, \mathsf{salt})$ for $i \in \{0,1\}$.
- Let $X_0 := \mathsf{F}_0(\mathsf{sd}, \mathsf{salt}_0),\ X_1 := \mathsf{F}_1(\mathsf{sd}, \mathsf{salt}_1)$.
- For $i \in [2, D]$, define $X_{b_1,\ldots,b_{i-1},0} = \mathsf{F}_0(\mathsf{F}_{b_{i-1}}(X_{b_1,\ldots,b_{i-1}}), \mathsf{salt}_0),\ X_{b_1,\ldots,b_{i-1},1} = \mathsf{F}_1(\mathsf{F}_{b_{i-1}}(X_{b_1,\ldots,b_{i-1}}), \mathsf{salt}_1)$ where $b_j \in \{0,1\}$ for all $j \in [1, i-1]$.
- The formula is generalized to compute the leaf of the tree as follows:

  For each $i \in [0, n-1]$, bit-decompose i as $\sum_{j=1}^{D} 2^{j-1} \cdot i_j$ for $i_j \in \{0,1\}$ then:

$$\begin{aligned} X_i = X_{i_1,\ldots,i_D} &= \mathsf{F}_{i_D}(\mathsf{F}_{i_{D-1}}(X_{i_1,\ldots,i_{D-1}}), \mathsf{salt}_{i_D}) \\ &= \mathsf{F}_{i_D}(\mathsf{F}_{i_{D-1}}(\ldots(\mathsf{F}_{i_1}(\mathsf{sd}_{i_1}, \mathsf{salt}_{i_1}), \mathsf{salt}_{i_{D-1}}), \mathsf{salt}_{i_D}) \end{aligned}$$

  To formalize, the value for each leaf $i \in [0, n-1]$ is denoted as:

$$\begin{aligned} \mathsf{PPRF}_{\mathsf{sd}}(\mathsf{salt}, i) &= \mathsf{F}_{i_D}\left(\mathsf{PPRF}_{\mathsf{sd}}\left(\mathsf{salt}, i_{1,\ldots,D-1}\right), \mathsf{salt}\right) \\ &= \mathsf{F}_{i_D}\left(\mathsf{F}_{i_{D-1}}\left(\ldots\left(\mathsf{F}_{i_1}(\mathsf{sd}, \mathsf{salt}), \mathsf{salt}\right), \mathsf{salt}\right)\right. \end{aligned}$$

  where $i_{1,\ldots,k} = \sum_{j=1}^{k} 2^{k-j} i_j$ for any $k \in [1, D]$.
- The co-path $\mathsf{CoPath}(i)$ for each $i = \sum_{j=1}^{D} 2^{j-1} \cdot i_j \in [0, n-1]$ is defined as follows:

$$\mathsf{CoPath}(i) = \mathsf{CoPath}(X_{i_1,\ldots,i_D}) = \{X_{\bar{i}_1}, X_{i_1,\bar{i}_2}, \ldots, X_{i_1,\ldots,\bar{i}_D}\}$$

  Formalizing, then:

$$\mathsf{CoPath}_{\mathsf{sd}}(\mathsf{salt}, i) = \mathsf{PPRF}_{\mathsf{sd}}\left(\mathsf{salt}, i_{1,\ldots,\bar{j}}\right)_{j=1,\ldots,D}$$

  where $i_{1,\ldots,\bar{k}} = \sum_{j=1}^{k-1} 2^{k-j}.i_j + \bar{i}_k$ for any $k \in [1, D]$.

Protocol 12: New construction $\mathsf{PPRF}(\mathsf{sd}, \mathsf{salt}, 2^D)$ *of Puncturable PRF*

Aalysis of the construction in the ideal cipher model

Theorem 6.2.1 (PPRF security). *Assume that* $\mathsf{PRG} = (\mathsf{F}_0, \mathsf{F}_1)$ *with* $\mathsf{F}_b : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$ *is an* (N, τ)*-instance* (t, ϵ)*-secure length-doubling PRG. Then the construction* $\mathsf{PPRF}(\mathsf{sd}, \mathsf{salt}, 2^D)$ *described in 12 is an* (N, τ)*-instance strongly* $(t, D \cdot \epsilon)$*-secure PPRF with input domain* $[2^D]$ *and punctured key domain* $(\{0,1\}^{\lambda})^D$.

Proof. The proof proceeds through a sequence of hybrids, each relying on the (N, τ)-instance security of $\mathsf{F}_0, \mathsf{F}_1$. For each leaf $i^{(j,e)} \in \{0,1\}^D$ in each tree $\mathsf{PPRF}(\mathsf{sd}_{j,e}, \mathsf{salt}_j, 2^D)$, the value assigned to this leaf $i^{(j,e)}$ is denoted $X_{i_1^{(j,e)},\ldots,i^{(j,e)}D}$. The secret path from the root $(\mathsf{sd}j, e, \mathsf{salt}_j)$ to the leaf $i^{(j,e)}$ is the tuple of intermediate nodes $\{X_{i_1^{(j,e)}}, X_{i_1^{(j,e)},i_2^{(j,e)}}, \ldots, X_{i_1^{(j,e)},\ldots,i_D^{(j,e)}}\}$.
Experiment 0 $(\mathsf{Exp}^{\mathbf{0}})$. All trees of the N instances are generated through the scheme described in Protocol 12, which is executed at each level to produce the leaves of the next level. Specifically, for each $j \leq N, e \leq \tau$, the construction of the (j,e)-th tree begins with a random master $(\mathsf{sd}_{j,e}, \mathsf{salt}_j)$ and uses F_0 and F_1 across all 2^D levels to generate the right and left children.
Experiment 1 $(\mathsf{Exp}^{\mathbf{1}})$. This experiment is identical to the previous one except for the first level of each tree. For all $j = 1, \ldots, N, e \leq \tau$, the leaves at the first level $(X_{1^{(j,e)}}, X_{0^{(j,e)}})$ are not generated using $\mathsf{F}_0, \mathsf{F}_1$ but are instead randomly sampled. Since $\mathsf{F}_0, \mathsf{F}_1$ is an (N, τ)-instance (t, ϵ)-secure PRG, the following holds:

$$|\Pr[\mathsf{Exp}^0(\lambda) = 1] - \Pr[\mathsf{Exp}^1(\lambda) = 1]| \leq \epsilon(\lambda)$$

Experiment 2 $(\mathsf{Exp}^{\mathbf{2}})$. The difference from the previous experiment lies in the second level of each tree: all the leaves $(X_{i_1^{(j,e)},0}, X_{i_1^{(j,e)},1})$ previously computed using F_0 and F_1 are now randomly chosen for each $j = 1, \ldots, N, e \leq \tau$. Using the secure property of F_0 and F_1, the following holds:

$$|\Pr[\mathsf{Exp}^1(\lambda) = 1] - \Pr[\mathsf{Exp}^2(\lambda) = 1]| \leq \epsilon(\lambda)$$

As can be deduced, traversing along the secret path of each tree, the mechanism of replacing the two leaves $(X_{i_1^{(j,e)},\ldots,i_{k-1}^{(j,e)},0}, X_{i_1^{(j,e)},\ldots,i_{k-1}^{(j,e)},1})$ at each level $k \in [1, D]$ by uniformly random values can continue through the entire depth D of the tree. This process results in D experiments, and applying the same hypothesis about the security of F_0 and F_1, the following is obtained:

$$|\Pr[\mathsf{Exp}^{i-1}(\lambda) = 1] - \Pr[\mathsf{Exp}^{i}(\lambda) = 1]| \leq \epsilon(\lambda)$$

for all $i = 1, \ldots, D$. Furthermore, during this traversal, all values on the *co-path* to the leaves $i^{(j,e)}$ are simultaneously replaced by uniformly random values.
Experiment D $(\mathsf{Exp}^{\mathbf{D}})$. In the final experiment, all nodes on the co-path to $i^{(j,e)}$ as well as the leaf $i^{(j,e)}$ are chosen uniformly at random for $j = 1$ to N and $e = 1$ to τ. The final

bound, which concludes the proof, is:

$$|\Pr[\mathsf{Exp}^0(\lambda) = 1] - \Pr[\mathsf{Exp}^D(\lambda) = 1]| \leq D \cdot \epsilon(\lambda),$$

□

The crux of the analysis is now to demonstrate that the PRG is (N, τ)-instance (t, ε)-secure for a suitable choice of N, τ, t, ε. Since the PRG explicitly uses a block cipher, reliance on the random permutation model is no longer feasible. Instead, security is proven in the *ideal cipher model*, where each key $K \in \{0,1\}^\lambda$ defines a truly random permutation π_K, and all parties are given oracle access to π_K and π_K^{-1} for all K. The attacker's running time t is measured as its number of queries q to the oracles. Using Patarin's H-coefficient technique, it is formally proven that the construction is an (N, τ)-instance (q, ε)-secure PRG for any N up to $2^{\lambda-1}$, with $\varepsilon \leq \frac{4\tau\cdot\lambda}{\ln\lambda} \cdot \frac{q}{2^\lambda}$, where the term $4\tau\lambda/\ln\lambda$ can be replaced by 8τ when $N \leq 2^{\lambda/2}$. The analysis is non-trivial, with the bound derived from a careful study of the influence of collisions among seeds on the adversarial advantage. The number of such collisions is bounded using standard lemmas on the maximum load of a bin when $2N$ balls are thrown into 2^λ bins.

6.2.3 A Multi-Instance PRG in the Ideal Cipher Model

In this section, the construction of a multi-instance PRG in the ideal cipher model is described. The construction itself is not entirely new but is a tweak on a construction of [GKW+20]. The work of [GKW+20] provides a construction of PPRF in the random permutation model, obtained by applying the GGM reduction to the following "Davies-Meyer" construction of a length-doubling PRG $G : x \to (\pi_0(x) \oplus x, \pi_1(x) \oplus x)$, where (π_0, π_1) are pseudorandom permutations. The PRG is proven secure in the random permutation model (in the analysis, all parties are given oracle access to π_0, π_1, and their inverses). A simple yet effective observation is that the most efficient instantiation of this construction implements the permutations π_0, π_1 by fixing two keys (K_0, K_1) and defining $\pi_b := E_{K_B}$, where E_{K_B} is a *block cipher* (such as AES). This leads to the following idea: instead of fixing the keys (K_0, K_1), sample them randomly and use them as a salt for the PRG in the multi-instance setting. The candidate multi-instance PRG becomes $G = (\mathsf{F}_0, \mathsf{F}_1) : (x, \mathsf{salt}) \to (E_{\mathsf{salt}_0}(x) \oplus x, E_{\mathsf{salt}_1}(x) \oplus x)$. The formal construction is given in Figure 6.1. While the high-level intuition is straightforward, the formal analysis is considerably more involved. The remainder of this section is devoted to a formal proof that the above construction is an (N, τ)-instance (t, ε)-secure PRG, for parameters $(N, \tau, t, \varepsilon)$ specified later. The proof is in the *ideal cipher model*: in this model, each key $K \in \{0,1\}^\lambda$ defines an independent uniformly random permutation π_K. All parties have access to an oracle which, on input $(0, K, x)$, outputs $\pi_K(x)$, and on input $(1, K, y)$, outputs $\pi_K^{-1}(y)$.

Definition 6.2.4 (Ideal Cipher Oracle). *For every $K \in \{0,1\}^\lambda$, let $\pi_K : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a uniformly random permutation over $\{0,1\}^\lambda$. The ideal cipher oracle $\mathcal{O}_\pi$ is defined as follows:*

- *On input* $(x, K) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$, *outputs* $\pi_K(x)$.
- *On input* (inv, x, K), *outputs* $\pi_K^{-1}(x)$.

Parameters:

- For each $K \in \{0,1\}^\lambda$, $\pi_K : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a uniformly random permutation.

Construction:

- Sample $\mathsf{salt} \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$. Parse $\mathsf{salt} := (K_0, K_1)$.
- $\mathsf{F}_b : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ is defined as $\mathsf{F}_b(\mathsf{sd}, \mathsf{salt}_b) = \pi_{K_b}(\mathsf{sd}) \oplus \mathsf{sd}$ for $b \in \{0,1\}$ and $\mathsf{sd} \in \{0,1\}^\lambda$.

Figure 6.1: Multi-instance PRG $\mathsf{F}_0, \mathsf{F}_1$ *in the ideal cipher model*

Theorem 6.2.2. *Let* $\mathsf{F}_0, \mathsf{F}_1$ *be the functions defined in Figure 6.1. Let* q *be the number of queries to the oracle* $\mathcal{O}_\pi$. *Then* $(\mathsf{F}_0, \mathsf{F}_1)$ *is an* (N, τ)*-instance* (q, ϵ)*-secure* PRG *in the ideal cipher model (where the parties are given oracle access to* $\mathcal{O}_\pi$ *from Definition 6.2.4), where*

$$\varepsilon \le f_N(\lambda) \cdot q \cdot \left(\frac{1}{2^{\lambda-1}} + \frac{1}{2^\lambda - q}\right) + \frac{4\tau N}{2^{2\lambda}},$$

for some function f_N *such that if* $N \le 2^{\lambda-1}$, $f_N(\lambda) \le \frac{3\tau\lambda \cdot \ln 2}{\ln\lambda + \ln\ln 2}$, *and if* $N \le 2^{\lambda/2}$, $f_N(\lambda) \le 4\tau$.

Proof. Fix a number of instances N and a number of repetitions τ. A distinguisher $\mathcal{D}$ is considered that receives $(\mathsf{salt}_i, (y_{i,e})_{e\le\tau})_{i\le 2N}$ according to either the real world experiment $\mathsf{Exp}_{\mathcal{D}}^{\mathsf{rw\text{-}prg}}$ or the ideal world experiment $\mathsf{Exp}_{\mathcal{D}}^{\mathsf{iw\text{-}prg}}$ of Definition 6.2.3, interacts with the ideal cipher oracle $\mathcal{O}_\pi$, and outputs a guess b. Let q be a bound on the number of queries of $\mathcal{D}$ to $\mathcal{O}_\pi$. To simplify the discussion, it is assumed that the $N \cdot \tau$ seeds $(\mathsf{sd}^{(1,e)}, \cdots, \mathsf{sd}^{(N,e)})_{e\le\tau}$ are also sampled (but not used) in the experiment $\mathsf{Exp}_{\mathcal{D}}^{\mathsf{iw\text{-}prg}}$. The notation salt_i is written as (K_0^i, K_1^i).

Reformulating the experiment. Now, sample $(\mathsf{sd}^{(1,e)}, \cdots, \mathsf{sd}^{(N,e)})_{e\le\tau}$ and pairs of keys $(K_0^i, K_1^i)_{i\le N}$. If $N \cdot \tau$ is large, with a high probability there will be some collisions among the seeds. Let $M \le N \cdot \tau$ denote the number of distinct seeds. To simplify the analysis, the seeds and the keys are reordered and renamed as follows:

- $\mathsf{sd}_1, \ldots, \mathsf{sd}_M$ are the M distinct seeds from the set of $N \cdot \tau$ sampled seeds $\mathsf{sd}^{(j,e)}$. For each seed sd_i, define $S_i \subseteq \{0,1\}^\lambda$ to be the set of indices such that $K \in S_i$ if there is an index (j, e) such that $\mathsf{sd}^{(j,e)} = \mathsf{sd}_i$ and either $K = K_0^j$ or $K = K_1^j$ (that is, sd_i was

sampled at least once together with a salt that contains K). Note that S_i corresponds to all keys K such that π_K is queried on sd_i in $\mathsf{Exp}^{\mathsf{rw\text{-}prg}}_{\mathcal{D}}$.

- For each π_K, define $S'_K := \{i : K \in S_i\} \subseteq [M]$ to be the set of indices of seeds that will be queried to π_K.

With the above notations, the distinguisher $\mathcal{D}$ receives the sets $S_1, \cdots, S_M$, and for each $i \leq M$, it gets either $\pi_K(\mathsf{sd}_i) \oplus \mathsf{sd}_i$ for all $K \in S_i$ (experiment $\mathsf{Exp}^{\mathsf{rw\text{-}prg}}_{\mathcal{D}}$), or a set of random values $(y_{K,i})_{K \in S_i}$ (experiment $\mathsf{Exp}^{\mathsf{iw}}_{\mathcal{D}}$). These alternative experiments only differ from the original experiments if it happens that two seeds $\mathsf{sd}^{(i,e)}, \mathsf{sd}^{(j,f)}$ collide, and two of their keys (K^i_0, K^i_1) and (K^j_0, K^j_1) also collide: in this case, the original experiments would return distinct values y in the ideal world, but identical values in the real world, making them trivially distinguishable. However, the probability of this even happening is very small:

$$\Pr[\exists (i,e) \neq (j,f), \mathsf{sd}^{(i,e)} = \mathsf{sd}^{(j,f)} \wedge \exists (b_i, b_j) \in \{0,1\}^2, K^i_{b_i} = K^j_{b_j}] \leq \frac{4N \cdot \tau}{2^{2\lambda}}.$$

Condition on this event not happening, the new experiments become perfectly equivalent to the original experiments. A flag is therefore raised if the above condition occurs, the process aborts if a flag is raised, and the focus shifts to bounding the distinguishing advantage in these new experiments.

Bounding the size of S'_K. The maximum size of S'_K for any K is now bounded. A standard lemma on the maximum load of a bin when tossing m balls into n bins is needed:

Lemma 6.2.1 (balls-and-bins). *Consider tossing m balls into n bins. For $m \leq n$, denoting* max_load *as the maximum number of balls that end up in any single bin, we have*

$$\Pr\left[\mathsf{max\_load} \geq \frac{3 \ln n}{\ln \ln n}\right] \leq \frac{1}{n}.$$

By definition, the maximum size of S'_K is reached for the permutation π_K that is invoked on the largest number of distinct seeds. A tight upper bound on this number follows from a simple balls-and-bins analysis: each time τ new seeds $(\mathsf{sd}^{(i,e)})_{e \leq \tau}$ are sampled, two keys (K^i_0, K^i_1) are sampled, which can be viewed as throwing two balls to two random bins, sampled randomly from 2^λ possible bins. After N steps of this experiment (hence after throwing $2N$ balls at random), denoting max_load the maximum load of any bin, $\tau \cdot$max_load is an upper bound on $\max_K |S'_K|$.[3] We get:

[3] This is a very tight upper bound: because $|S'_K|$ counts only *distinct* seeds, it overcounts whenever it happens that a new seed $\mathsf{sd}^{(j,e)}$ is sampled that collides with one of the previous seeds ($\mathsf{sd}^{(jne)} = \mathsf{sd}^{(i,f)}$ for some $i < j$) and K^j_0 or K^j_1 also collides with one of the two keys (K^i_0, K^i_1). However, the chance that this happens is at most $\tau \cdot N / 2^{2\lambda}$.

Claim 6.2.1. *Whenever* $2N \leq 2^\lambda$*, the maximum load* $\max_K |S'_K|$ *is bounded by* $\frac{3\tau \cdot \ln 2^\lambda}{\ln \ln 2^\lambda}$ *with probability* $1 - 2^{-\lambda}$*. Furthermore, if* $2N \leq 2^{\lambda/2}$*,* $\max_K |S'_K|$ *is bounded by* 4τ *with probability* $1 - 2^{-\lambda}$*.*

The first part of the claim follows directly from the balls-and-bins lemma 6.2.1. The last part of the claim follows from the fact that when $2N \leq 2^{\lambda/2}$, the probability of having 4 balls in any given bin is at most $1/2^{2\lambda}$, and the claim follows by a union bound over the 2^λ bins.

Bounding the advantage of $\mathcal{D}$. The next step focuses on bounding the advantage of $\mathcal{D}$ in distinguishing the real world and the ideal world experiments. Below, the *transcript* of the interaction of $\mathcal{D}$ in the experiments is formally defined:

Definition 6.2.5 (Transcript)**.** *A transcript of $\mathcal{D}$'s interaction is defined as*

$$Q = ((y_{i,j})_{i \leq M, j \in S_i}, Q_\pi, (\mathsf{sd}_i)_{i \leq M})$$

where $Q_\pi = (z, j, \pi_j(z))$ records all queries/answers to/from the permutation oracle $\mathcal{O}_\pi$ (queries for the inverse of permutation can be considered as $(\pi_b^{-1}(z), b, z)$. Note that $(\mathsf{sd}_i)_{i \leq M}$ is included to facilitate the analysis but is not available to the distinguisher $\mathcal{D}$: in the real-world, $(\mathsf{sd}_i)_{i \leq M}$ are used to compute $(y_{i,j})_{i \leq M, j \in S_i}$, whereas in the ideal-world, $(y_{i,j})_{i \leq M, j \in S_i}$ are sampled uniformly at random from $\{0,1\}^\lambda$.

*A transcript Q is called **attainable** for some fixed $\mathcal{D}$ if there exist some oracles $\mathcal{O}_\pi$ such that the interaction of $\mathcal{D}$ with those oracles would lead to transcript Q.*

In the game of distinguishing between the ideal world and the real world, the distinguishing advantage is expressed as:

$$\mathsf{Adv}(\mathcal{D}^{\mathcal{O}}_\pi) = |\mathsf{Pr}_{\mathsf{rw}}[\mathcal{D}^{\mathcal{O}_\pi} = 1] - \mathsf{Pr}_{\mathsf{iw}}[\mathcal{D}^{\mathcal{O}_\pi} = 1]|.$$

The proof relies on Patarin's H-coefficient technique [Pat09; CS14], summarized below. The H-coefficient theorem helps in bounding the advantage of the distinguisher by categorizing the set of attainable transcripts into "good" and "bad" transcripts:

Theorem 6.2.3 (H-coefficient)**.** *Fix a distinguisher $\mathcal{D}$. Let $\mathcal{T}$ denote the set of attainable transcripts Q, and let $\mathsf{Pr}_{\mathsf{rw}}$ and $\mathsf{Pr}_{\mathsf{iw}}$ represent the probabilities of events in the real and ideal world, respectively. Let $\mathcal{T}_{\mathsf{bad}}$ denote a set of "bad" transcripts, and $\mathcal{T}_{\mathsf{good}} = \mathcal{T} \setminus \mathcal{T}_{\mathsf{bad}}$ be the set of "good" transcripts. Suppose that:*

- $\mathsf{Pr}_{\mathsf{iw}}[Q \in \mathcal{T}_{\mathsf{bad}}] \leq \nu$.
- $\left|\frac{\mathsf{Pr}_{\mathsf{rw}}[Q]}{\mathsf{Pr}_{\mathsf{iw}}[Q]} - 1\right| \leq \mu$ *for all* $Q \in \mathcal{T}_{\mathsf{good}}$.

Then $\mathsf{Adv}(\mathcal{D}^{\mathcal{O}}_\pi) \leq \nu + \mu$.

One key insight of the H-coefficient technique is that the ratio $\frac{\mathsf{Pr}_{\mathsf{rw}}[Q]}{\mathsf{Pr}_{\mathsf{iw}}[Q]}$ corresponds to the ratio between the probability that the real-world oracles are consistent with Q and the probability that the ideal-world oracles are consistent with Q. Denote $\Pr[\text{RW is consistent with Q}]$ and $\Pr[\text{IW is consistent with Q}]$ as $\mathsf{Pr}_{\mathsf{rw}}(Q)$ and $\mathsf{Pr}_{\mathsf{iw}}(Q)$, respectively. Then:

$$\forall Q \in \mathcal{T}_{\mathsf{good}},\ \frac{\mathsf{Pr}_{\mathsf{rw}}[Q]}{\mathsf{Pr}_{\mathsf{iw}}[Q]} = \frac{\mathsf{Pr}_{\mathsf{rw}}(Q)}{\mathsf{Pr}_{\mathsf{iw}}(Q)}.$$

The goal is now to use the H-coefficient theorem to prove Theorem 6.2.2. Define $\mathcal{T}_{\mathsf{bad}}$ and $\mathcal{T}_{\mathsf{good}}$ based on distinct seeds and permutations:

- $\mathcal{T}_{\mathsf{bad}}$ contains transcripts $Q = ((y_{i,K})_{i\le M, K\in S_i}, Q_\pi, (\mathsf{sd}_i)_{i\le M}) \in \mathcal{T}$ such that:
  - $\exists(\mathsf{sd}_i, K, *) \in Q_\pi$ with $K \in S_i$.
  - $\exists(*, K, \mathsf{sd}_i \oplus y_{i,K})$ with $K \in S_i$.
- $\mathcal{T}_{\mathsf{good}} = \mathcal{T} \setminus \mathcal{T}_{\mathsf{bad}}$.

Bounding $\mathsf{Pr}_{\mathsf{iw}}[Q \in \mathcal{T}_{\mathsf{bad}}]$. Denote $|Q_\pi| = q = \sum_{K\in\{0,1\}^\lambda} q_K$, where $q_K := |Q_{\pi_K}| := |\{(*, K, *) \in Q_\pi\}|$ for $K \in \{0,1\}$. In the ideal-world, $(\mathsf{sd}_i)_{i\le M}$ are independent of $((y_{i,K})_{i\le M, K\in S_i})$, and we have:

$$\begin{aligned}
\mathsf{Pr}_{\mathsf{iw}}[Q \in \mathcal{T}_{\mathsf{bad}}] &\le \sum_{K\in\{0,1\}^\lambda} (\mathsf{Pr}_{\mathsf{iw}}[\exists(\mathsf{sd}_i, K, *) \in Q_\pi \mid i \in S'_K] \\
&\qquad + \mathsf{Pr}_{\mathsf{iw}}[\exists(*, K, y_{i,K} \oplus \mathsf{sd}_i) \in Q_\pi \mid i \in S'_K]) \\
&= \sum_{K\in\{0,1\}^\lambda} \frac{2q_K \cdot |S'_K|}{|2^\lambda|} = \frac{1}{2^{\lambda-1}} \cdot \sum_{K\in\{0,1\}^\lambda} q_K \cdot |S'_K| \\
&\le \frac{1}{2^{\lambda-1}} \cdot q \cdot \max_K |S'_K|.
\end{aligned}$$

Bounding $\mathsf{Pr}_{\mathsf{rw}}[Q]/\mathsf{Pr}_{\mathsf{iw}}[Q]$ for $Q \in \mathcal{T}_{\mathsf{good}}$. First, compute the probability $\mathsf{Pr}_{\mathsf{iw}}(Q)$ that the ideal-world oracle is consistent with Q. Denote $\left((\mathsf{sd}'_i)_{i\le M}, (y'_{i,K})_{i\le M, K\in S_i}, (\pi_K)_{K\in\{0,1\}^\lambda}\right)$ as an arbitrary setting of the ideal world experiment, where $(\mathsf{sd}'_i)_{i\le M}, (y_{i,K})_{i\le M, K\in S_i}$ are sampled as in $\mathsf{Exp}^{\mathsf{iw}}_{\mathcal{D}}(\lambda)$, and $\pi_K : \{0,1\}^\lambda \to \{0,1\}^\lambda$ are fixed random permutations. Let $\pi \vdash Q_\pi$ denote the event that the permutation π is consistent with the queries/answers in Q_π. Write $(\pi_K)_K \vdash Q_\pi$ to indicate that random permutations π_K are consistent with all queries in the transcript Q_π. Since in the ideal world all these values are sampled independently,

denoting $p_\pi = \Pr_{\pi_K}[(\pi_K)_{K\in\{0,1\}^\lambda} \vdash Q_\pi)]$, we have:

$$\begin{aligned}
&\mathsf{Pr}_{\mathsf{iw}}(Q) = \\
&\Pr_{\mathsf{sd}'_i, y'_{i,K}, \pi_K}[(\mathsf{sd}'_i = \mathsf{sd}_i) \ \wedge \ (\forall K \in S_i,\ y_{i,K} = y'_{i,K}) \ \wedge \ ((\pi_K)_{K\in\{0,1\}^\lambda} \vdash Q_\pi)] \\
&= \Pr_{\mathsf{sd}'_i}[\mathsf{sd}'_i = \mathsf{sd}_i] \ \cdot \ \Pr_{y'_{i,K}}[\forall K \in S_i,\ y_{i,K} = y'_{i,K}] \ \cdot p_\pi \\
&= \left(\frac{1}{2^\lambda}\right)^M \cdot \left(\frac{1}{2^\lambda}\right)^{\sum_{i=1}^M |S_i|} \cdot \prod_{K\in\{0,1\}^\lambda} \frac{1}{(2^\lambda)_{q_K}} \\
&= 2^{-\lambda\cdot\sum_{K\in\{0,1\}^\lambda}|S'_K|} \cdot \prod_{K\in\{0,1\}^\lambda} \frac{1}{(2^\lambda)_{q_K}} \cdot 2^{-\lambda\cdot M}.
\end{aligned}$$

where for $1 \le b \le a$, $(a)_b := a\cdot(a-1)\cdot(a-2)\cdots(a-b+1)$. Note that the last equality comes from the fact that $\sum_{i=1}^M |S_i| = \sum_{K\in\{0,1\}^\lambda} |S'_K|$.
We next compute the probability $\mathsf{Pr}_{\mathsf{rw}}(Q)$ that the real-world oracle is consistent with Q. Denote $((\mathsf{sd}'_i)_{i\le M}, (y'_{i,K})_{i\le M, K\in S_i}, (\pi_K)_{K\in\{0,1\}^\lambda})$ as a setting of the real world. The main difference is that $(y'_{i,K})_{i\le M,K\in S_i}$ are now dependent on $(\mathsf{sd}'_i)_{i\le M}$. Denoting $p_\pi = \Pr_{\pi_K}[(\pi_K)_{K\in\{0,1\}^\lambda} \vdash Q_\pi)]$, we have:

$$\begin{aligned}
&\mathsf{Pr}_{\mathsf{rw}}(Q) = \\
&\Pr[(\mathsf{sd}'_i = \mathsf{sd}_i) \ \wedge \ (\forall K \in S_i,\ y_{i,K} = y'_{i,K}) \ \wedge \ ((\pi_K)_{K\in\{0,1\}^\lambda} \vdash Q_\pi)] \\
&= \Pr_{\mathsf{sd}'_i}[(\mathsf{sd}'_i = \mathsf{sd}_i) \ \wedge \ (\forall K \in S_i,\ y_{i,K} = \pi_K(\mathsf{sd}_i) \oplus \mathsf{sd}_i)] \cdot p_\pi \\
&= \Pr_{\mathsf{sd}'_i}[(\mathsf{sd}'_i = \mathsf{sd}_i) \ \wedge \ (\forall i \in S'_K,\ y_{i,K} = \pi_K(\mathsf{sd}_i) \oplus \mathsf{sd}_i)] \cdot p_\pi \\
&= \Pr_{\pi_K}\left[\ y_{i,K} = \pi_K(\mathsf{sd}_i) \oplus \mathsf{sd}_i \quad (\pi_K)_{K\in\{0,1\}^\lambda} \vdash Q_\pi \ \right] \cdot p_\pi \cdot \Pr_{\mathsf{sd}'_i}[\mathsf{sd}'_i = \mathsf{sd}_i] \\
&= \frac{1}{2^{\lambda\cdot M}} \cdot \prod_{K\in\{0,1\}^\lambda} \frac{1}{(2^\lambda)_{q_K}} \cdot \Pr_{\pi_K}\left[\ y_{i,K} = \pi_K(\mathsf{sd}_i) \oplus \mathsf{sd}_i \quad (\pi_K)_{K\in\{0,1\}^\lambda} \vdash Q_\pi \ \right].
\end{aligned}$$

Since $Q \in \mathcal{T}_{\mathsf{good}}$, then $\nexists(\mathsf{sd}_i, K, *) \in Q_\pi$ with $K \in S_i$, and $\nexists(*, K, \mathsf{sd}_i \oplus y_{i,K}) \in Q_\pi$ with $K \in S_i$. This leads to:

$$\begin{aligned}
&\Pr_{\pi_K}\left[\ y_{i,K} = \pi_K(\mathsf{sd}_i) \oplus \mathsf{sd}_i \quad (\pi_K)_{K\in\{0,1\}^\lambda} \vdash Q_\pi \ \right] \\
&= \prod_{K\in\{0,1\}^\lambda} \Pr_{\pi_K}[\pi_K(\mathsf{sd}_i) = y_{i,K} \oplus \mathsf{sd}_i] = \prod_{K\in\{0,1\}^\lambda} \frac{1}{(2^\lambda - q_K)_{|S'_K|}}.
\end{aligned}$$

Putting equations together, we obtain:

$$\mathsf{Pr}_{\mathsf{rw}}(Q) = \frac{1}{2^{\lambda \cdot M}} \cdot \prod_{K \in \{0,1\}^\lambda} \frac{1}{(2^\lambda)_{q_K}} \cdot \prod_{K \in \{0,1\}^\lambda} \frac{1}{(2^\lambda - q_K)_{|S'_K|}}$$

and eventually:

$$\begin{aligned} \forall Q \in \mathcal{T}_{\mathsf{good}},\ \frac{\mathsf{Pr}_{\mathsf{rw}}[Q]}{\mathsf{Pr}_{\mathsf{iw}}[Q]} = \frac{\mathsf{Pr}_{\mathsf{rw}}(Q)}{\mathsf{Pr}_{\mathsf{iw}}(Q)} &= \prod_{K \in \{0,1\}^\lambda} \frac{2^{\lambda \cdot \sum_{K \in \{0,1\}^\lambda} |S'_K|}}{(2^\lambda - q_K)_{|S'_K|}} \\ &= \prod_{K \in \{0,1\}^\lambda} \frac{2^{2N \cdot \lambda}}{(2^\lambda - q_K)_{|S'_K|}}. \end{aligned}$$

Distinguishing advantage. Equipped with the above calculations, we can finally bound the distinguishing advantage of $\mathcal{D}^{\mathcal{O}^\pi}$. To upper bound $\mathsf{Adv}(\mathcal{D}^{\mathcal{O}^\pi})$, we upper bound the ratio $\frac{\mathsf{Pr}_{\mathsf{rw}}[Q]}{\mathsf{Pr}_{\mathsf{iw}}[Q]}$, which translates to computing a lower bound on $\prod_{K \in \{0,1\}^\lambda} (2^\lambda - q_K)_{|S'_K|}$. Denote $K_{\mathsf{max}} \in K \in \{0,1\}^\lambda$ the index of the set among all $\{S'_K\}_{K \in \{0,1\}^\lambda}$ that has $\max_K |S'_K|$ elements. Then we have

$$\begin{aligned} \prod_{K \in \{0,1\}^\lambda} (2^\lambda - q_K)_{|S'_K|} &\geq \prod_{K \neq K_{\mathsf{max}}} (2^\lambda)^{|S'_K|} \cdot (2^\lambda - q)_{\max_K |S'_K|} \\ &= (2^\lambda)^{\sum_{K \in \{0,1\}^\lambda} |S'_K|} \cdot \frac{(2^\lambda - q)_{\max_K |S'_K|}}{(2^\lambda)^{\max_K |S'_K|}} \\ &= 2^{2N \cdot \lambda} \cdot \frac{(2^\lambda - q)_{\max_K |S'_K|}}{(2^\lambda)^{\max_K |S'_K|}} \geq 2^{2N \cdot \lambda} \cdot \left(\frac{2^\lambda - q}{2^\lambda} \right)^{\max_K |S'_K|}. \end{aligned}$$

$$\implies \frac{\mathsf{Pr}_{\mathsf{rw}}[Q]}{\mathsf{Pr}_{\mathsf{iw}}[Q]} \leq \left(\frac{2^\lambda}{2^\lambda - q} \right)^{\max_K |S'_K|} = \left(1 + \frac{q}{2^\lambda - q} \right)^{\max_K |S'_K|}.$$

The above yields

$$\frac{\mathsf{Pr}_{\mathsf{rw}}[Q]}{\mathsf{Pr}_{\mathsf{iw}}[Q]} \leq 1 + \frac{q \cdot \max_K |S'_K|}{2^\lambda - q}.$$

Then, using the H-coefficient theorem (Theorem 6.2.3), we get:

$$\mathsf{Adv}(\mathcal{D}^{\mathcal{O}^\pi}) = \frac{1}{2^{\lambda-1}} \cdot q \cdot \max_K |S'_K| + \frac{q \cdot \max_K |S'_K|}{2^\lambda - q}.$$

Plugging the bound on $|S'_K|$ from the claim finishes the proof. □ □

This result implies that the new multi-instance PPRF construction can serve as a drop-in replacement for previous (much slower) hash-based constructions, with only a small security loss of $4\tau D\lambda / \ln \lambda$ (or $8\tau D$ when the number of signature queries is limited to $2^{\lambda/2}$). For $D = 16$, $\tau = 8$, and $\lambda = 128$, this translates to a security loss of 14 bits for up to 2^{127}

queries or 10 bits for up to 2^{64} queries. Additionally, this loss can be reduced to 8 bits by only guaranteeing that the *expected* runtime of the adversary exceeds 2^{λ}.
An optimization that converts an (N, τ)-instance (t, ε)-secure PRG to a $(\tau \cdot N, 1)$-instance (t, ε)-secure PRG by using a pseudorandom generator to sample the τ salts $(\mathsf{salt}^{i,e})_{e \leq \tau}$ for a given instance from a global salt salt_i for each $i \leq N$ can be introduced: this reduces the security loss by a factor of τ, resulting in a loss of 5 bits for $D = 16$, $\tau = 8$, and $\lambda = 128$. This tradeoff is considered very reasonable given the benefits of using a significantly faster AES-based construction. Finally, it is possible to take into account an additional optimization that reduces the security loss to 3 bits, independently of D, by using a pseudorandom generator to generate $(\tau \cdot D)$ salts $(\mathsf{salt}^{i,e})e \leq \tau, i \leq D$ from a global salt salt and evaluating each *level* of each GGM tree with a distinct salt. In this approach, all salts are randomly sampled, leading to a negligible probability of collisions among $(\mathsf{salt}^{i,e}, \mathsf{sd}^{i,e})e \leq \tau, i \leq N$. It is conjectured that this variant can be proven secure with only a 1-bit loss in the ideal cipher model. Proving this conjecture is expected to require a considerably more intricate direct analysis of the full multi-instance PPRF in the ideal cipher model, bypassing the reduction to a multi-instance PRG again.

6.2.4 Application

The new PPRF is expected to find applications beyond MPCitH signatures. As a sample application, pseudorandom correlation functions [BCG+20a; BCG+22], which are used to efficiently generate correlated randomness in secure computation, are typically constructed using a large number of distributed point functions (DPFs). DPFs are very similar to GGM-style PPRFs, and the analysis is expected to extend almost immediately to multi-instance DPFs. According to the attack described in Section 6.2, the concrete security of PCFs using N copies of a DPF with a tree of depth D is $2^{\lambda}/(N \cdot D)$. In many scenarios, this represents a significant security loss. For example, using the PCF of [BCG+22] to generate 2^{30} degree-2 correlations requires $N = 664^2$ copies of a GGM tree of depth $\log_2(2^{30}/664)^2$ (a 2-dimensional GGM tree). Under the collision attack, this results in a concrete loss of 27 bits of security. Extending GGM PPRFs to the multi-instance setting using the described methodology could reduce the security loss to 5 bits without sacrificing efficiency.
The proof technique developed is also expected to improve the parameters of other schemes. For instance, the recent work [BCS24] leverages AES in a half-tree construction [GYW+23] to improve VOLEitH signatures. The scheme in [BCS24] uses a direct construction of a half-tree based on a circular collision-resistant (CCR) hash function. Due to the security limitation of CCR hash (to 128-bit blocks and key size) when using an AES-based instantiation, the size of tree leaves has been extended to $2\lambda = 256$ bits to achieve 128 bits of security. The techniques developed in the concrete security analysis of the multi-instance PPRF using the H-coefficient technique are expected to apply to the security proof in [BCS24], potentially eliminating the need to expand the size of the last layer.

6.3 The Improved Signature Scheme

This section introduces a new signature scheme based on the regular syndrome decoding assumption, obtained by improving the previous signature presented in Section 5.4.
Recall that, given $H \in \mathbb{F}_2^{k \times K}$ and $x \in \mathbb{F}_2^K$ as a w-regular vector with block size $\mathsf{bs} \leftarrow K/w$, the protocol described in Section 5.4 verifies that x is regular and satisfies $H \cdot x = y$ by leveraging the MPC-in-the-head paradigm. This reduces to checking:

(1) $\sum_{i=1}^{\mathsf{bs}} x_i = 1 \bmod \mathsf{bs}$ for $\mathbb{Z}_{\mathsf{bs}}$-shares of x,

(2) $H \cdot x = y$ for $\mathbb{F}_2$-shares of x.

As linear checks are "free" in MPCitH, the problem simplifies to designing a *sharing conversion* protocol to transform $\mathbb{F}_2$-shares into $\mathbb{Z}_{\mathsf{bs}}$-shares. This transformation is made efficient by leveraging precomputed mod-2 and mod-bs shares of the same random bit. However, the main efficiency bottleneck of the protocol stems from exactly the use of shares over $\mathbb{Z}_{\mathsf{bs}}$: because of that, the signature includes several (one for each of the τ repetitions of the basic proof) length-K vectors over $\mathbb{Z}_{\mathsf{bs}}$ (using a CRT trick, this can be reduced to $\mathbb{Z}_{\mathsf{bs}/2}$ whenever $\mathsf{bs}/2$ is odd and ≥ 3). This yields a $O(K \cdot \mathsf{bs})$ communication cost, which is (by a significant margin) the dominant cost of the protocol. To mitigate this cost, the block size bs was set to be the smallest possible value $\mathsf{bs} = 6$ (such that $\mathsf{bs}/2 = 3$). In turn, this forces to rely on RSD with very high weight $w = K/6$, which requires significantly increasing the parameters to compensate for the security loss.
The first observation is that all of these shortcomings can be eliminated at once by relying on an alternative share conversion approach. Because x is w-regular, it admits a compressed representation as a list of w integers in $[\mathsf{bs}]$, which indicates the position of the nonzero entry in each of the w unit vectors. Now, observe that if the parties hold shares of w integers $(i_1, \cdots, i_w)$ modulo bs, these can always be interpreted as representing some regular vector x; in other words, given such shares, condition (1) is satisfied by default. The crux of the protocol is a conversion procedure that turns shares of this compressed representation into shares modulo 2 of the "decompressed" regular vector (with which the parties can check the linear equation $H \cdot x = y$ for free). Furthermore, this share conversion can again be implemented very efficiently if the parties are given shares of pairs of the same random unit vector in compressed representation and in standard representation. Concretely, given an integer $r \in [\mathsf{bs}]$, let $\mathbf{e}_r$ denote the length-bs unit vector with a 1 at position r. Assume that the n parties, holding shares of some $i \in [\mathsf{bs}]$, are given shares of r modulo bs, and shares of $\mathbf{e}_r$ over $\mathbb{F}_2$. Consider the following simple protocol:

- All parties broadcast their shares of $z = i - r \bmod \mathsf{bs}$ and reconstruct z.
- All parties locally shift cyclically their share of $\mathbf{e}_r$ by z.

After this protocol, all parties end up with shares of the vector $\mathbf{e}_r$ shifted by z, which is denoted $\mathbf{e}_r \downarrow z$ (vectors are viewed as columns, hence the shift by z is downward). Observe that $\mathbf{e}_r \downarrow z = \mathbf{e}_r \downarrow (i - r) = (\mathbf{e}_r \uparrow r) \downarrow i = \mathbf{e}_{\mathsf{bs}} \downarrow i = \mathbf{e}_i$. As in Protocol 3, the prover generates w random pairs $(r, \mathbf{e}_r)$ and shares them between the virtual parties. To dispense with the

need to check that the pairs were honestly generated, the same strategy is relied upon, where the verifier samples a random permutation π of $[w]$ and instructs the prover to shuffle the pairs according to π before using them in the protocol. The high-level structure of the MPCitH-compiled zero-knowledge proof (without optimizations) is below:

MPCitH-compiled Zero-knowledge Proof of Knowledge

- **Parameters and input:** let (K, k, w) be parameters for the syndrome decoding problem, and let $\mathsf{bs} \leftarrow K/w$. The prover holds a w-regular witness $x \in [\mathsf{bs}]^w$ (in compressed representation) for the relation $H \cdot x = y$, where $H \in \mathbb{F}_2^{k \times K}$ and $y \in \mathbb{F}_2^k$ are public. Let n be the number of virtual parties.
- **Round 1:** the prover samples w pairs $(r_i, \mathbf{e}_{r_i})$ where $r_i \leftarrow\!\!\$\ [\mathsf{bs}]$. These pairs are denoted $(\mathbf{r}, \mathbf{e}_\mathbf{r})$. The prover generates n shares of $\mathbf{e}_\mathbf{r}$ (over $\mathbb{F}_2$) and of $x, \mathbf{r}$ (modulo bs) distributed between the virtual parties, and commits to the local state of each party.
- **Round 2:** the verifier samples and sends to the prover a random permutation $\pi \leftarrow\!\!\$\ \mathsf{Perm}(w)$. This is written as $\pi(\mathbf{r})$ (resp. $\pi(\mathbf{e}_\mathbf{r})$) for the vector $(r_{\pi(1)}, \cdots, r_{\pi(w)})$ (resp. $(\mathbf{e}_{r_{\pi(1)}}, \cdots, \mathbf{e}_{r_{\pi(w)}})$).
- **Round 3:** the prover executes the following protocol:
  - All parties reconstruct $\mathbf{z} = \mathbf{x} - \pi(\mathbf{r})$ and shift their shares of $\pi(\mathbf{e}_\mathbf{r})$, getting shares of $\pi(\mathbf{e}_\mathbf{r}) \downarrow \mathbf{z}$ (the shifting is done blockwise: each $\mathbf{e}_{r_{\pi(i)}}$ is cyclically shifted by z_i). Note that $\pi(\mathbf{e}_\mathbf{r}) \downarrow \mathbf{z} = \mathbf{e}_x$ (*i.e.* the "uncompressed" representation of the witness x).
  - All parties compute a share of $H \cdot (\pi(\mathbf{e}_\mathbf{r}) \downarrow \mathbf{z})$ and broadcast them. All parties check that the shares reconstruct to y.
- **Round 4:** the verifier picks $i \leftarrow\!\!\$\ [n]$ and challenges the prover to open the views of all parties except i.
- **Round 5:** the prover sends the $n - 1$ openings to the verifier, who checks that the views are consistent with the commitments, with each other, and with the output of the protocol being y.

The soundness of the scheme is $\varepsilon = \mathsf{p} + (1/n) \cdot (1 - \mathsf{p})$, where $\mathsf{p} = \mathsf{p}(K, k, w)$ is an upper bound on the probability (over the choice of the random permutation π) that a cheating prover, that commits in the first round to an incorrect witness (*i.e.* a compressed vector x^* such that $H \cdot \mathbf{e}_{x^*} \neq y$), manages to generate a valid MPC transcript (*i.e.* finds —possibly incorrect— pairs $(\mathbf{r}, \mathbf{u})$ such that $H \cdot (\pi(\mathbf{u}) \downarrow \mathbf{z}) = y$, where $\mathbf{z} = x^* - \pi(\mathbf{r})$). The crux of the analysis lies in computing a tight evaluation of p.
In the final signature, multiple optimizations are incorporated on top of this basic template, including the usual optimization of generating the shares in a tree-based fashion using the GGM puncturable pseudorandom function [KPT+13; BW13; BGI14; GGM86], but also the

more recent hypercube technique from [AGH+23], and a number of additional optimizations tailored to the scheme.
In terms of signature size, the dominant cost stems from the size of a share of x and of w pairs $(r, \mathbf{e}_r)$ (using standard optimizations, all shares except one can be compressed, hence the communication is dominated by the size of a single share, ignoring for now the number of repetitions of the identification scheme). The size of a share of x together with w pairs $(r, \mathbf{e}_r)$ is $2w \log \mathsf{bs} + K$ bits[4], whereas the size of x (now shared as a vector over $\mathbb{F}_2^K$) and of the pairs in 5.2.1 is $K \cdot (2 + \mathsf{bs}/2)$ bits. This directly incurs a significant reduction in the signature size. Furthermore, with this alternative conversion, using a very small block size is not advantageous anymore, which allows exploration of a much wider range of parameters, resulting in further savings.

6.3.1 Description of the signature scheme

The key generation algorithm (Protocol 13) randomly samples a syndrome decoding instance (H, y) with solution x . The signing algorithm with secret key $\mathsf{sk} = (H, y, x)$ and message $m \in \{0,1\}^*$ is described in Protocol 14. The verification algorithm with public key $\mathsf{pk} = (H, y)$ (matrix H can be computed from PRG with a random seed, the public key size is around 0.09kB), message $m \in \{0,1\}^*$, and signature σ, is described in Protocol 15. [5]

Key generation algorithm

Inputs: A security parameter λ.

1. Sample $\mathsf{sd} \leftarrow \{0,1\}^\lambda$;
2. Set $H \leftarrow \mathsf{PRG}(\mathsf{sd})$ with $H \in \mathbb{F}_2^{k \times K}$;
3. Sample $x \leftarrow\!\!\$\ [\mathsf{bs}]^w$ and set $y \leftarrow H \cdot \mathsf{Expand}(x)$ and $\mathsf{sk} \leftarrow (\mathsf{sd}, x)$.

Protocol 13: Key generation algorithm of the new signature scheme

Signing Algorithm

Inputs. A secret key sk and a message $m \in \{0,1\}^*$.

Initialization. Parse sk as (sd, x).

- Let $H \leftarrow \mathsf{PRG}(\mathsf{sd})$ and $y \leftarrow H \cdot \mathsf{Expand}(x)$; // $H \in \mathbb{F}_2^{k \times K}$ is a (pseudo)random

[4] As in the previous signature, this number is multiplied by a number τ of repetition, but since it is the same in both works, it is ignored in this discussion for simplicity.

[5] For readability, the description of the signing and verification algorithms ignores an optimization that slightly reduces the signature size, but significantly complexifies the description. This optimization, already presented in 5.2.2.1, leverages the regular structure of x to reduce its side from $K = w \cdot \mathsf{bs}$ to $w \cdot (\mathsf{bs} - 1) = K - w$ bits by sharing only the $\mathsf{bs} - 1$ first entries $(u_1, \cdots, u_{\mathsf{bs}-1})$ of each block of $\mathbf{u}^e$, since the last one can be reconstructed as $\bigoplus u_i \oplus 1$.

matrix in systematic form.

- Sample $(K_0, K_1) \leftarrow\!\!\$\ \{0,1\}^\lambda \times \{0,1\}^\lambda$. Set $\mathsf{salt} \leftarrow (K_0, K_1)$.

Phase 1. For each iteration $e \in [\tau]$:

- Sample $\mathsf{sd}^e \leftarrow\!\!\$\ \{0,1\}^\lambda$;
- For $d = 1$ to D, set $(X^e_{d,0}, R^e_{d,0}, U^e_{d,0}) \leftarrow (0,0,0) \in [\mathsf{bs}]^w \times [\mathsf{bs}]^w \times \{0,1\}^K$;
- Set $x^e_n \leftarrow x$, $u^e_n \leftarrow 0$, and $r^e \leftarrow 0$;
- **For** $i = 1$ **to** $n-1$:
  1. Compute $\mathsf{sd}^e_i \leftarrow \mathsf{F}_{\mathsf{salt}}(\mathsf{sd}^e, i)$; // Can be computed efficiently by always storing the path to the current node: to move from i to $i+1$, start from the closest ancestor of $i+1$ in the path to leave i.
  2. Set $\mathsf{state}^e_i \leftarrow \mathsf{sd}^e_i$;
  3. $(x^e_i, r^e_i, u^e_i, \mathsf{com}^e_i) \leftarrow \mathsf{PRG}(\mathsf{sd}^e_i)$; // $(x^e_i, r^e_i, u^e_i, \mathsf{com}^e_i) \in [\mathsf{bs}]^w \times [\mathsf{bs}]^w \times \{0,1\}^K \times \{0,1\}^\lambda$.
  4. $x^e_n \leftarrow x^e_n - x^e_i \bmod \mathsf{bs}$, $u^e_n \leftarrow u^e_n \oplus u^e_i$, and $r^e \leftarrow r^e + r^e_i \bmod \mathsf{bs}$;
  5. For all $d \le D$ such that $i[d] = 0$, set: // $i[d]$ is the d-th bit of the integer i.
     - $X^e_{d,0} \leftarrow X^e_{d,0} + x^e_i \bmod \mathsf{bs}$;
     - $R^e_{d,0} \leftarrow R^e_{d,0} + r^e_i \bmod \mathsf{bs}$;
     - $U^e_{d,0} \leftarrow U^e_{d,0} \oplus u^e_i$;
- **On node** n**:**
  1. Compute $\mathsf{sd}^e_n \leftarrow \mathsf{F}_{\mathsf{salt}}(\mathsf{sd}^e, n)$;
  2. Compute $r^e_n \leftarrow \mathsf{PRG}(\mathsf{sd}^e_n)$;
  3. $r^e \leftarrow r^e + r^e_n \bmod \mathsf{bs}$, $u^e \leftarrow \mathsf{Expand}(r^e)$, and $u^e_n \leftarrow u^e_n \oplus u^e$; // The $(x^e_i)_i$ form n pseudorandom shares of $x \in [\mathsf{bs}]^w$, the $(r^e_i)_i$ form n pseudorandom shares of $r^e \in [\mathsf{bs}]^w$, and the $(u^e_i)_i$ form n pseudorandom shares of $u^e = \mathsf{Expand}(r^e) \in \{0,1\}^K$.
  4. Define $\mathsf{aux}^e_n \leftarrow (x^e_n, u^e_n)$;
  5. Set $\mathsf{state}^e_n \leftarrow \mathsf{aux}^e_n || \mathsf{sd}^e_n$ and $\mathsf{com}^e_n \leftarrow \mathsf{H}(\mathsf{state}^e_n)$.

Phase 2.
1. $h_1 \leftarrow \mathsf{H}_1(m, \mathsf{salt}, \mathsf{com}^1_1, \cdots, \mathsf{com}^1_N, \cdots, \mathsf{com}^\tau_1, \cdots, \mathsf{com}^\tau_N)$; // Accumulate the commitments inside the hash rather than storing and hashing all at once.
2. $\pi^e_{\{e \in \tau\}} \leftarrow \mathsf{PRG}_1(h_1)$. // $\pi^e \in \mathsf{Perm}([w])$.

Phase 3. For each iteration $e \in [\tau]$:

1. $z^e \leftarrow x - \pi^e(r^e) \bmod \mathsf{bs}$;
2. For $d = 1$ to D, set:

- $y^e_{d,0} \leftarrow H \cdot \mathsf{Shift}(\pi^e(U^e_{d,0}), z^e)$ and $y^e_{d,1} \leftarrow y^e_{d,0} \oplus y$;
- $z^e_{d,0} \leftarrow X^e_d - \pi^e(R^e_{d,0}) \mod \mathsf{bs}$ and $z^e_{d,1} \leftarrow z^e - z^e_{d,0} \bmod \mathsf{bs}$.

Phase 4. 1. $h_2 \leftarrow \mathsf{H}_2(m, \mathsf{salt}, h_1, (y^e_{d,b}, z^e_{d,b})_{d\le D, b\in\{0,1\}, e\le\tau})$;

2. Set $(b^e_1, \cdots b^e_D)_{e\le\tau} \leftarrow \mathsf{PRG}_2(h_2)$ and let $i^e \leftarrow \sum_{d=1}^{D} b^e_d \cdot 2^{d-1}$.

Phase 5. Output $\sigma = \left(\mathsf{salt}, h_1, h_2, (\mathsf{CoPath}_{\mathsf{salt}}(i^e, \mathsf{sd}^e), z^e, \mathsf{com}^e_{i^e}, \mathsf{aux}^e_n)_{e\le\tau}\right)$. // aux^e_n is not included if $i^e = n$.

Protocol 14: Signing algorithm of the new signature scheme

Verification Algorithm

Inputs. A public key $\mathsf{pk} = (H, y)$, a message $m \in \{0,1\}^*$ and a signature σ.

1. Split the signature as follows:

$$\sigma = \left(\mathsf{salt}, h_1, h_2, (\mathsf{CoPath}_{\mathsf{salt}}(i^e, \mathsf{sd}^e), z^e, \mathsf{com}^e_{i^e}, \mathsf{aux}^e_n)_{e\le\tau}\right);$$

2. Recompute $\pi^e_{\{e\in\tau\}}$ where $\pi^e \in \mathsf{Perm}([w])$ via a pseudorandom generator using h_1;
3. Recompute $(b^e_1, \cdots b^e_D)_{e\le\tau}$ via a pseudorandom generator using h_2 and define $i^e \leftarrow \sum_{d=1}^{D} b^e_d \cdot 2^{d-1}$;
4. For each iteration $e \in [\tau]$,
   - For $d = 1$ to D:
     - Denote $b = 1 - b^e_d$;
     - Set $(X^e_{d,b}, R^e_{d,b}, U^e_{d,b}) \leftarrow (0,0,0) \in [\mathsf{bs}]^w \times [\mathsf{bs}]^w \times \{0,1\}^K$;
     - For each $i \neq i^e$:
       * Recompute sd^e_i from the $\mathsf{CoPath}_{\mathsf{salt}}(i^e, \mathsf{sd}^e)$;
       * If $i \neq n$, recompute $(x^e_i, r^e_i, u^e_i, \mathsf{com}^e_i) \leftarrow \mathsf{PRG}(\mathsf{sd}^e_i)$; else, parse aux^e_n as (x^e_n, u^e_n), and compute $r^e_n \leftarrow \mathsf{PRG}(\mathsf{sd}^e_n)$;
       * If $i[d] = b$, update:
         - $X^e_{d,b} \leftarrow X^e_{d,b} + x^e_i \bmod \mathsf{bs}$;
         - $R^e_{d,b} \leftarrow R^e_{d,b} + r^e_i \bmod \mathsf{bs}$;
         - $U^e_{d,b} \leftarrow U^e_{d,b} \oplus u^e_i$;
     - Recompute $(y^e_{d,b}, z^e_{d,b})$ by simulating the Phase 3 of the signing algorithm as below:
       - $y^e_{d,b} \leftarrow H \cdot \mathsf{Shift}(\pi^e(U^e_{d,b}), z^e)$;
       - $z^e_{d,b} \leftarrow X^e_{d,b} - \pi^e(R^e_{d,b}) \mod \mathsf{bs}$;

- Recompute $(y^e_{d,1-b}, z^e_{d,1-b})$ as below:
  - $y^e_{d,1-b} \leftarrow y^e_{d,b} \oplus y$;
  - $z^e_{d,1-b} \leftarrow z^e - z^e_{d,b} \bmod \mathsf{bs}$;

5. Check if $h_1 \leftarrow \mathsf{H}_1(m, \mathsf{salt}, \mathsf{com}^1_1, \cdots, \mathsf{com}^1_N, \cdots, \mathsf{com}^\tau_1, \cdots, \mathsf{com}^\tau_N)$;
6. Check if $h_2 \leftarrow \mathsf{H}_2(m, \mathsf{salt}, h_1, (y^e_{d,b}, z^e_{d,b})_{d\le D, b\in\{0,1\}, e\le\tau})$;
7. Output ACCEPT if both conditions are satisfied.

Protocol 15: Verification algorithm of the new signature scheme

Theorem 6.3.1. *Assume that* PPRF *is a* (q_s, τ)*-instance* $(t, \epsilon_{\mathsf{F}})$*-secure PPRF, that* PRG *is a* (q_s, τ)*-instance* $(t, \epsilon_{\mathsf{PRG}})$*-secure PRG, and that any adversary running in time* t *has at advantage at most* ϵ_{SD} *against the regular syndrome decoding problem. Model the hash functions* $\mathsf{H}_1, \mathsf{H}_2$ *as random oracles with output of length* 2λ*-bit and the pseudorandom generator* PRG_2 *as a random oracle. Then chosen-message adversary against the signature scheme 14, running in time* t*, making* q_s *signing queries, and making* q_1, q_2, q_3 *queries, respectively, to the random oracles* $\mathsf{H}_1, \mathsf{H}_2$ *and* PRG_2*, succeeds in outputting a valid forgery with probability*

$$\Pr[\text{Forge}] \le \frac{q_s\,(q_s + q_1 + q_2 + q_3)}{2^{2\lambda}} + \epsilon_{\mathsf{F}} + \varepsilon_{\mathsf{PRG}} + \epsilon_{\mathsf{SD}} + \Pr[X + Y = \tau] + \varepsilon_{\mathsf{G}} + \frac{1}{2^\lambda},$$

where $\epsilon = \mathsf{p} + \frac{1}{N} - \frac{\mathsf{p}}{N}$*, with* $\mathsf{p} = 4/B$ *and* $\varepsilon_{\mathsf{G}} = \varepsilon_{\mathsf{G}}(K, k, w, B)$ *is* $\Pr[(H, y) \notin \mathsf{GOOD}_B]$*, which is defined on Lemma 6.4.2,* $X = \max_{\alpha\in Q_1}\{X_\alpha\}$ *and* $Y = \max_{\beta\in Q_2}\{Y_\beta\}$ *with* $X_\alpha \sim \mathsf{Binomial}(\tau, \mathsf{p})$ *and* $Y_\beta \sim \mathsf{Binomial}\left(\tau - X, \frac{1}{N}\right)$ *where* Q_1 *and* Q_2 *are sets of all queries to oracles* H_1 *and* H_2*.*

Computing the bound p from Theorem 6.3.1 requires a dedicated combinatorial analysis, extensively covered in Section 6.4.1. The proof of Theorem 6.3.1 is deferred to Section 6.4.2.

6.3.2 Parameters selection

In this section, the process of selecting parameters for the new signature scheme is explained. The first goal is to pick parameters that minimize the number of repetitions τ of the underlying identification scheme, since this parameter has a large impact on the signature size. Concretely, as in previous chapter, τ is chosen such that the cost of the forgery attack on the Fiat-Shamir-compiled signature is at least 2^{128}, where cost is given by the formula below (which comes from the attack of Kales and Zaverucha [KZ20a]):

$$\mathsf{cost} = \min_{\tau_1,\tau_2:\tau_1+\tau_2=\tau} \left\{ \frac{1}{\sum_{i=\tau_1}^{\tau} \binom{\tau}{i} \mathsf{p}^i (1-\mathsf{p})^{\tau-i}} + N^{\tau_2} \right\}. \tag{6.1}$$

It is observed that setting $B = 2^{-134}$ in $\mathsf{p} = 4/B$ suffices to guarantee that τ is always the

smallest possible for any given number of leaves $n = 2^D$ (*i.e.* reducing p further does not reduce τ). The choice of the number of leaves, 2^D, is a tradeoff parameter: larger values of D yield smaller signature size, at the expense of a larger runtime.

Finding a bound on k. A crucial aspect of the parameter selection process is that the combinatorial analysis in Section 6.4.1 only guarantees that with very high probability, for any $(\mathbf{u}, x^*) \in \mathsf{PN}_B \times [\mathsf{bs}]^w$ where $\mathbf{u}$ is not regular, $\pi(\mathbf{u}) \downarrow x^*$ will not be a valid solution $\mathbf{v}$ to $H \cdot \mathbf{v} = y$. However, it says nothing about vectors $\mathbf{u}$ outside PN_B, that is, vectors with low permutation number $\mathsf{pn}(\mathbf{u}) \leq B$. Therefore, RSD parameters must be selected such that, with overwhelming probability, there will not be *any* solution $\mathbf{v}$ to $H \cdot \mathbf{v} = y$ of the form $\mathbf{v} = \mathbf{u} \downarrow x^*$ for $\mathbf{u} \in \mathbb{F}_2^K \setminus \mathsf{PN}_B$. Since the set $X = \{\mathbf{v} \in \mathbb{F}_2^K \ : \ \exists \mathbf{u} \in \mathbb{F}_2^K \setminus \mathsf{PN}_B, \exists x^* \in [\mathsf{bs}]^w, \mathbf{v} = \mathbf{u} \downarrow x^*\}$ is defined, to guarantee that there will not be any solution $\mathbf{v} \in X$ to $H \cdot \mathbf{v} = y$, it suffices to pick $\log_2 k \geq |X| + \lambda$. This follows from a standard "Gilbert-Varshamov-style" analysis: when sampling a random instance $(H, y = H \cdot x)$ of the RSD problem, the expected number of solutions in X (beyond x) is

$$\mathop{\mathbb{E}}_{H,x}\left[|\{x' \ : \ H \cdot x' = H \cdot x \ \wedge \ x' \in X\}|\right] = \sum_{\substack{x' \neq x \\ x' \in X}} \Pr_{H,x}[H \cdot x' = H \cdot x] = \frac{|X| - 1}{2^k},$$

and the conclusion is reached with a Markov bound. To choose k, a bound on $|X|$ is used:

Lemma 6.3.1. *Let $\mathsf{P}_{\mathsf{i,w}}$ denote the set of* integer partitions *of w in i parts,* i.e., *the set of all tuples $(k_1, \cdots, k_i)$ with $0 < k_1 \leq k_2 \leq \cdots \leq k_i \leq w$ such that $\sum_{j=1}^{i} k_j = w$. Let $\mathsf{T_B}$ denote the function such that $\mathsf{T_B}(x) = x$ when $x \leq B$, and $\mathsf{T_B}(x) = 0$ when $x > B$. Then*

$$|X| \leq \mathsf{bs}^w \cdot \sum_{i=1}^{L} \left(\binom{L}{i} \cdot i! \cdot \sum_{(k_1, \cdots, k_i) \in \mathsf{P}_{\mathsf{i,w}}} \mathsf{T_B}\left(\frac{w!}{\prod_{j=1}^{i}(k_j)!} \right) \right),$$

where L is (using the Euler totient ϕ and denoting $a|b$ for "a divides b"):

$$L = \frac{1}{\mathsf{bs}} \cdot \sum_{\substack{i \leq \mathsf{bs} \\ i \ odd}} \sum_{d \mid \gcd(\mathsf{bs}-i, i)} \phi(d) \cdot \binom{\mathsf{bs}/d}{i/d}.$$

Proof. The proof of Lemma 6.3.1 follows from a counting argument, detailed below. The bs^w possible vectors $x^* \in [\mathsf{bs}]^w$ are enumerated, and all possible $\mathbf{u}$ with $\mathsf{pn}(\mathbf{u}) \geq B$ are counted. To count the latter, the process proceeds in steps:

Counting the number of distinct blocks. The number L of possible distinct blocks is computed. A loose upper bound would be $L \leq 2^{\mathsf{bs}}$ (since a block is a vector in $\mathbb{F}_2^{\mathsf{bs}}$). However, because all possible shifts x^* of the w blocks are already enumerated, only the number of distinct blocks *up to cyclic shift* is counted. In combinatorics, this amounts to counting the number of length-bs necklaces with two colors. Additionally, because of the optimization

given in Section 5.2.2.1 where the last entry of each block is fixed such that all entries of a block XOR to 1, only necklaces with an odd number of ones are enumerated. The formula for L in Lemma 6.3.1 is a direct application of Pólya's enumeration theorem [Red27], a classical theorem on the combinatorics of necklaces.

Counting the number of vectors. For $i = 1$ to L, the number of vectors which have exactly i distinct blocks is counted. There are $\binom{L}{i}$ ways to select the i distinct blocks out of L possible blocks. Since each vector has w blocks in total, all partitions of the integer w in exactly i parts $0 < k_1 \leq k_2 \leq \cdots \leq k_i \leq w$ are enumerated, where k_j denotes the number of copies of the j-th block from the selection. Because *ordered* partitions are enumerated, the i selected blocks are ordered by number of copies; hence, multiplied by $i!$ to account for all possible configurations of number of copies (this is a slightly loose upper bound, since some partitions may have equal numbers $k_j = k_{j+1}$: the right value would be to multiply by the factorial of the number of distinct integers in $(k_1, \cdots, k_j)$, but this is ignored for simplicity). Then, having fixed a choice of i specific distinct blocks and the numbers $(k_1, \cdots, k_i)$ of copies of each block, there are $w!/\prod_{j=1}^{i}(k_j)!$ distinct blockwise permutations of (this is the standard combinatorial formula for counting multisets). Eventually, since only vectors whose permutation number is at most B are kept, only in the count are included the vectors for which $w!/\prod_{j=1}^{i}(k_j)! \leq B$ (this is the purpose of the threshold function T_B in the formula). This yields the formula stated in Lemma 6.3.1.

Computing $|X|$. It remains to compute explicitly the formula of Lemma 6.3.1. A Python script [6] is used to perform the calculation. A small nontriviality is that enumerating over all integer partitions of w (which is around 120) would be very slow. Fortunately, it is observed that the condition $w!/\prod_{j=1}^{i}(k_j)! \leq B$, together with the bound L on i, impose a sharp bound on the value of k_i: a quick calculation shows that $k_i \geq w/2$ is needed to be such that $\binom{w}{k_i} \leq B$. Given this bound on k_i, all remaining possible values of k_i are enumerated, and the number of partitions of $w - k_i$ into $i - 1$ parts is computed to obtain the rest of the partition. The script used to compute this bound is available at https://github.com/ElianaCarozza/Short-Signatures-from-Regular-Syndrome-Decoding-in-the-Head/blob/main/script.py. □

Finding (K, w). To find the RSD parameters (K, k, w), an iterative process is used: a choice of K, w is fixed, and the value of k is computed as $|X| + 128$ (note that X depends on (K, w)), using the script to compute the bound on $|X|$ from Lemma 6.3.1. Then, the estimator implemented in state-of-the-art cryptanalysis of [ES23] (which improves over previous cryptanalysis from 5.5 is relied upon to compute (an estimate of) the bit security of the instance obtained against all known attacks on RSD. If the bit security is below 128, K is increased by 1 and the process restarts (each time, the parameters for a list of weight parameters w are also computed, since the impact of w on the proof size is slightly subtle). Eventually, after settling for a choice of (K, k, w), it is checked that the probability bound of Lemma 6.4.2 is overwhelming (with the chosen parameters, it is always above $1 - 2^{-200}$).

[6] https://github.com/ElianaCarozza/Faster-Signatures-from-MPC-in-the-Head/blob/main/script.py

6.3.2.1 Concrete Parameters and Implementation Outlined below are a few parameter sets for different values of $D \in \{8, \cdots, 17\}$. For all values of D, the smallest signature size was achieved by setting $K = 1736$, $k = 960$, $w = 217$, $\mathsf{bs} = 8$.

D	τ	$\lvert\sigma\rvert$	signing time	verification time
8	16	7.8 kB	0.64 ms	0.56 ms
9	15	7.6 kB	0.88 ms	0.78 ms
10	13	6.8 kB	1.02 ms	0.94 ms
11	12	6.5 kB	1.40 ms	1.32 ms
12	11	6.1 kB	2.13 ms	2.05 ms
13	10	5.7 kB	3.56 ms	3.47 ms
15	9	5.4 kB	13.1 ms	13.0 ms
16	8	4.9 kB	23.9 ms	23.7 ms

Table 6.4: Signature size and signing time for various values of D, using the parameter set $K = 1736$, $k = 960$, $w = 217$, $\mathsf{bs} = 8$ with the AES-PPRF+ implementation. All timings are computed on one core of an Intel Core i7 processor 14700KF at 3.4 GHz frequency.

The signature scheme was implemented in C with three versions: a version with a tightly optimized folding (AES-PPRF+), a version with classical folding (AES-PPRF), and a version with classical folding using SHA instead of AES (on top of a proof-of-concept implementation with prior parameters, all of those versions provided in the artifact). The results in Table 3 are from the AES-PPRF+ implementation. It should be noted that the implementation did not use any optimizations such as batching, vectorization, or bit slicing, and an optimized implementation could likely achieve significantly faster runtime. The AES-NI instruction set was used to implement the multi-instance PPRF and the multi-instance PRG from Section 6.2.1. All experiments were run on one core of an Intel Core i7 processor 14700KF at 3.4 GHz frequency. The following optimization flags were used during compilation: `-O3 -lm -march=native`.
To ensure a fair comparison with SDitH, their benchmarking framework was used to obtain performance results. The overall implementation can be found in the artifacts of the conference (https://artifacts.iacr.org/).

6.4 Analysis

Although the high-level strategy –shuffling the random pairs– is the same as in the previous signature, the security analysis is *entirely* different from the one presented in Section 5.3.3. Shuffling the prover-generated correlated randomness is a highly non-generic technique, where each new protocol requires a new and dedicated combinatorial analysis.[7] The crux of the proof lies in bounding the success probability of a cheating adversary $\mathcal{A}$ in the following game:

[7] To give a sense of how specific the analysis of Section 5.3.3 was, not only does it work only for the specific type of pairs: it works exclusively for $\mathsf{bs} = 6$, corresponding to pairs of bits shared modulo 2 and modulo 3.

- $\mathcal{A}$ holds a vector $x^* \in [\mathsf{bs}]^w$ and chooses $\mathbf{r} \in [\mathsf{bs}]^w$ and $\mathbf{u} \in \mathbb{F}_2^K$, such that $\mathbf{u}$ is *not* a regular vector.
- A uniformly random permutation π is sampled from $\mathsf{Perm}(w)$.
- $\mathcal{A}$ wins iff $H \cdot (\pi(\mathbf{u})\downarrow(x^* - \pi(\mathbf{r}) \bmod \mathsf{bs})) = y$.

Given a bound on $\mathcal{A}$'s winning probability in this game, the rest of the proof follows in a relatively standard way and is similar to previous security proofs of code-based signature schemes in the MPCitH paradigm, such as the one presented in the previous chapter. Above, note that for any vector $\mathbf{s} \in [\mathsf{bs}]^w$, $\pi(\mathbf{u})\downarrow\mathbf{s}$ is a regular vector if and only if $\mathbf{u}$ is a regular vector. Note also that whether x^* is actually a correct witness or not (*i.e.* whether $H \cdot \mathbf{e}_{x^*}$) does not matter: as long as $\mathbf{u}$ is regular, if $\mathcal{A}$ wins the game above, then an extractor can recover a valid regular solution $\pi(\mathbf{u})\downarrow(x^* + \mathbf{r} \bmod \mathsf{bs})$ to the syndrome decoding problem (hence $\mathcal{A}$ "knew" a solution to the problem in the first place). Eventually, note that

$$\pi(\mathbf{u})\downarrow(x^* - \pi(\mathbf{r}) \bmod \mathsf{bs}) = \pi(\mathbf{u}\uparrow\mathbf{r})\downarrow x^*,$$

hence, the game above simplifies to the following: $\mathcal{A}$ chooses $x^* \in [\mathsf{bs}]^w$ and $\mathbf{u} \in \mathbb{F}_2^K \setminus \mathsf{Reg}_w$, and wins iff $H \cdot (\pi(\mathbf{u})\downarrow x^*) = y$ holds over the choice of a random permutation π.

Eliminating spurious solutions. An immediate issue with the above game is that an adversary *might* win with a very high probability if the system of equations $H \cdot x = y$ admits solutions that are mostly invariant by blockwise permutation. Concretely, assume that there exists a vector $\mathbf{u}^*$ which satisfies $H \cdot \mathbf{u}^* = y$, and such that $\mathbf{u}^*$ is not a regular vector, yet $\mathbf{v}^*$ is a concatenation of w *identical* vectors from $\mathbb{F}_2^{\mathsf{bs}}$. If this happens, then there is an easy winning strategy: $\mathcal{A}$ sets $\mathbf{u} \leftarrow \mathbf{u}^*$ and $x^* \leftarrow 0^w$. Since $H \cdot (\pi(\mathbf{u})\downarrow x^*) = H \cdot \pi(\mathbf{u}) = H \cdot \mathbf{u}^* = y$, $\mathcal{A}$ is guaranteed to win. More generally, if $H \cdot x = y$ admits a solution $\mathbf{u}$ whose blocks are *mostly* identical, then the equation $H \cdot \pi(\mathbf{u}^*) = y$ has a relatively large chance to hold simply because $\pi(\mathbf{u}^*)$ has a relatively large chance to be equal to $\mathbf{u}^*$.

Setting up some notations. Given a vector $\mathbf{u}$, let $\mathsf{pn}(\mathbf{u})$ denote $|\{\pi(\mathbf{u}) \mid \pi \in \mathsf{Perm}([w])\}|$. That is, $\mathsf{pn}(\mathbf{u})$ is the number of distinct vectors in $\mathbb{F}_2^K$ which can be obtained by shuffling $\mathbf{u}$ blockwise; $\mathsf{pn}(\mathbf{u})$ is called the *permutation number* of $\mathbf{u}$. Then, given a bound B, PN_B is defined as $\{\mathbf{u} \mid \mathsf{pn}(\mathbf{u}) > B\}$, the set of vectors with a large permutation number. Let X denote the set $\{\mathbf{v} \in \mathbb{F}_2^K \ : \ \exists \mathbf{u} \in \mathbb{F}_2^K \setminus \mathsf{PN}_B, \exists x^* \in [\mathsf{bs}]^w, \mathbf{v} = \mathbf{u} \downarrow x^*\}$. The set X captures exactly the possible spurious solutions: it contains the vectors $\mathbf{v}$ such that there exists some choice of the shift x^* such that $\mathbf{v}\uparrow x^*$ has a small permutation number ($\mathsf{pn}(\mathbf{v} \uparrow x^*) \leq B$). Denoting $\mathsf{Ker}(H) \oplus y$ the solutions to $H \cdot x = y$, if there is a vector $\mathbf{v} \in X \cap \mathsf{Ker}(H) \oplus y$, then $\mathcal{A}$ can pick $\mathbf{u}, x^*$ such that $\mathbf{v} = \mathbf{u}\downarrow x^*$ with $\mathsf{pn}(\mathbf{u}) \leq B$. This guarantees that with probability at least $1/B$, a random permutation π will satisfy $\pi(\mathbf{u}) = \mathbf{u}$, hence $H \cdot (\pi(\mathbf{u})\downarrow x^*) = H \cdot (\mathbf{u}\downarrow x^*) = H \cdot \mathbf{v} = y$.

Sampling highly-injective instances. Fix some bound B. To eliminate spurious solutions in X, which an adversary could use to win with probability at least $1/B$, parameters

(K, k, w) are chosen such that when sampling the regular syndrome decoding instance $(H, y = H \cdot x)$ (for some $x \in \mathsf{Reg}_w$), it holds with probability $1 - 1/2^\lambda$ that the *only* element of X that also belongs to $\mathsf{Ker}(H) \oplus y$ is the w-regular solution x. It follows from a standard analysis that this is the case as soon as $\log_2 k \geq \log_2 |X| + \lambda$.
To select k, a tight upper bound on $|X|$ is computed (see Lemma 6.3.1). Counting the number of elements of X is not entirely straightforward since the count is performed "up to some blockwise shift," but a closed formula can be established using known bounds for counting k-necklaces (*i.e.* bitstrings counted up to cyclic shifts) by leveraging Pólya's enumeration theorem [Red27]. Given the formula, a short Python program[8] is used to compute explicitly the bound on $|X|$ and select a suitable parameter k (for a fixed choice of K, w).
This also faces some challenges: the formula of Lemma 6.3.1 requires summing binomial coefficients over all *integer partitions* of the weight parameter w (*i.e.*, the number of tuples of distinct positive integers that sum to w). Because w is around 120, its number of integer partitions is too large to simply enumerate. With some careful consideration, many of these partitions can be eliminated from the counting procedure, and this observation is leveraged to reduce the runtime of the program.

Bounding the success probability. The focus is now shifted to the crux of the analysis: showing that if $\mathcal{A}$ picks $(\mathbf{u}, x^*)$ where $\mathsf{pn}(\mathbf{u}) > B$, then their probability of winning the game is at most $O(1/B)$ over the choice of the permutation π. What makes the analysis challenging is that in principle, it could be that some vector $\mathbf{u}$ has a high permutation number, yet *many of its permutations belong to* $\mathsf{Ker}(H) \oplus y$. The core technical component of the analysis is a proof that with very high probability over the choice of a random syndrome decoding instance (H, y), it will simultaneously hold for *all* vectors $\mathbf{u}$ with $\mathsf{pn}(\mathbf{u}) > B$ that for any choice of shift x^*, $\Pr_\pi[H \cdot (\pi(\mathbf{u}) \downarrow x^*) = y] \leq 4/B$. To state the result formally, a "good" syndrome decoding instance is defined below:

Definition 6.4.1 (GOOD_B). *Given a bound B, GOOD_B is defined as the set of syndrome decoding instances $(H, y) \in \mathbb{F}_2^{k \times K} \times \mathbb{F}_2^k$ such that for every $\mathbf{u} \in \mathsf{PN}_B \setminus \mathsf{Reg}_w$ and for all $x^* \in [\mathsf{bs}]^w$, $\Pr_{\pi \leftarrow\$ \mathsf{Perm}_w}[H \cdot (\pi(\mathbf{u}) \downarrow x^*) = y] \leq 4/B$.*

The main technical result of the analysis is stated below:

Lemma 6.4.1 (Most syndrome decoding instances are good)**.**

$$\Pr_{H,y}[(H, y) \in \mathsf{GOOD}_B] > 1 - \binom{2B}{5} \cdot \frac{2^{K+1}}{B \cdot 2^{3k}} \cdot \left(10 + \frac{(K/w)^w}{2^k}\right).$$

To parse the above, the reader can consider that $(K/w)^w \ll 2^k$ will hold the selected parameters, hence the probability that $(H, y) \in \mathsf{GOOD}_B$ is of the order of $1 - B^4 \cdot 2^{K-3 \cdot k}$. For concreteness, the reader can think of K as being around 1550, k as being around 820, w being around 200, and B as being around 70, resulting in the above being around $1 - 2^{-630}$.

[8] https://github.com/ElianaCarozza/Faster-Signatures-from-MPC-in-the-Head/blob/main/script.py

Key intuition. The main idea of the proof is outlined as follows. Given a vector $\mathbf{u}$ with $\mathsf{pn}(\mathbf{u}) = N$, fix some ordering $\mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(N)}$ of its distinct blockwise permutations, and let $x^* \in [\mathsf{bs}]^w$ denote some shift. Sample a random matrix $H \leftarrow\!\!\$\ \mathbb{F}_2^{k\times K}$, a random regular vector $x \leftarrow\!\!\$\ \mathsf{Reg}_w$, and set $y \leftarrow H \cdot x$. Let $(\mathbf{v}_1, \cdots, \mathbf{v}_N) \leftarrow ((\mathbf{u}^{(1)} \downarrow x) \oplus x, \cdots, (\mathbf{u}^{(N)} \downarrow x) \oplus x)$ (note that $H \cdot \mathbf{v}_i = 0$ iff $H \cdot (\mathbf{u}^{(i)} \downarrow x^*) = y$). Observe that the $\mathbf{v}_i$ are random variables, but they are set independently of H (since x is sampled independently from H). Then, for any subset S of t *linearly independent* vectors $\mathbf{v}_i$, it holds that

$$\Pr_{H\leftarrow\!\!\$\,\mathbb{F}_2^{k\times K}}[H \cdot \mathbf{v}_i = 0 \text{ for all } i \in S] = 2^{-k\cdot t}.$$

In other words, whenever the $\mathbf{v}_i$'s are linearly independent, the binary random variables X_i equal to 1 if $H \cdot \mathbf{v}_i = 0$ are independent. Building upon this observation, the following is shown: fix an arbitrary subset S of five indices. Then

- S contains a size-3 linearly independent subset with probability 1, and
- S contains a size-4 linearly independent subset, except with probability at most $10 \cdot (K/w)^{-w}$.

Together with the previous bound on the probability that $H \cdot \mathbf{v}_i = 0$ for linearly independent vectors, this yields a probability bound of $10 \cdot (K/w)^{-w}/2^{3\cdot k} + 1/2^{4\cdot k}$ that $H \cdot \mathbf{v}_i = 0$ for all $i \in S$. To see why this bound holds, observe that:

- The $\mathbf{v}_i$ are pairwise distinct and nonzero by construction (because $\mathbf{u}$ is assumed to be nonregular, so $\pi(\mathbf{u}) \downarrow x^*$ is never $\mathbf{0}$, and the $\mathbf{u}^{(i)}$ are distinct by definition).
- If e.g. $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ are linearly dependent, they therefore need to satisfy $\mathbf{v}_1 \oplus \mathbf{v}_2 \oplus \mathbf{v}_3 = \mathbf{0}$. But then, $\mathbf{v}_1 \oplus \mathbf{v}_2 \oplus \mathbf{v}_4 \neq \mathbf{0}$ (otherwise, one would have $\mathbf{v}_3 = \mathbf{v}_4$, contradicting the fact that the vectors are pairwise distinct). Hence, it is guaranteed to find a size-3 independent subset of vectors in S.
- By the same reasoning, S necessarily contains a 4-tuple of $\mathbf{v}_i$'s that does not XOR to 0, say, $(\mathbf{v}_1, \cdots, \mathbf{v}_4)$ (since if both $(\mathbf{v}_1, \cdots, \mathbf{v}_4)$ and $(\mathbf{v}_1, \cdots, \mathbf{v}_3, \mathbf{v}_5)$ XOR to 0, then $\mathbf{v}_4 = \mathbf{v}_5$). Then, either $(\mathbf{v}_1, \cdots, \mathbf{v}_4)$ is linearly independent (in which case the process is complete, since a 4-independent subset is found), or it must contain a size-3 subset that XORs to 0.
- For any subset of 3 $\mathbf{v}_i$'s, the probability that they XOR to 0 is at most $(K/w)^{-w}$. This follows from the fact that the $\mathbf{v}_i$'s are equal to $(\mathbf{a} \oplus x, \mathbf{b} \oplus x, \mathbf{c} \oplus x)$ for some *fixed* vectors $(\mathbf{a}, \mathbf{b}, \mathbf{c})$, and a uniformly random regular vector $x \in [\mathsf{bs}]^w$. But then, $\mathbf{v}_1 \oplus \mathbf{v}_2 \oplus \mathbf{v}_3 = 0$ rewrites to $\mathbf{a} \oplus \mathbf{b} \oplus \mathbf{c} = x$, which happens with probability at most $\mathsf{bs}^{-w} = (K/w)^{-w}$ over the choice of x.

Since there are 10 size-3 subsets of S, the bound follows. To summarize, a vector $\mathbf{u}$ with $\mathsf{pn}(\mathbf{u}) = N > B$ and a shift x^* were fixed, and it was shown that for every size-5 subset S of $[N]$, the probability that $H \cdot (\mathbf{u}^{(i)} \downarrow x^*) = y$ holds simultaneously for all $i \in S$ is at most $10 \cdot (K/w)^{-w}/2^{3\cdot k} + 1/2^{4\cdot k}$.

A careful union bound. To finish the proof of Lemma 6.4.1, it remains to compute a union bound over all possible vectors $\mathbf{u}$, shifts x^*, and size-5 subsets S. However, a quick calculation shows that a naive union bound does not suffice: first, the number of subsets is $\binom{N}{5}$, but since only $N > B$, the permutation number of $\mathbf{u}$, is known, it can only be bounded by $w!$, which is far too large. Second, the number of vectors $\mathbf{u}$ is 2^K, which is also too large for the union bound to yield a nontrivial result.
This issue is overcome by providing a more careful union bound. First, the distinct blockwise permutations of $\mathbf{u}$, $(\mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(N)})$, are divided into size-$B$ blocks of vectors. The previous bound is applied to all size-5 subsets inside each *block* of vectors, which reduces the factor resulting from the union bound to $(N/B) \cdot \binom{B}{5}$. This suffices to guarantee that in each size-B block, at most 4 vectors $\mathbf{v}_i$ can simultaneously satisfy $H \cdot \mathbf{v}_i = 0$, hence guaranteeing a success probability for $\mathcal{A}$ of at most $4/B$ over the random choice of π. Second, instead of enumerating over all vectors $\mathbf{u}$, enumeration is performed over all *equivalence classes* of vectors $\mathbf{u}$ which generate the same list $(\mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(N)})$. Each equivalence class contains exactly N vectors, and all equivalence classes are disjoint, saving a factor N this way from the union bound. Eventually, the union bound is finished by summing over all possible values of $N = \mathsf{pn}(\mathbf{u})$ from $B+1$ to $w!$. This finishes the proof of Lemma 6.4.1.

6.4.1 Combinatorial Analysis

In this section, bounds are provided on the probability that a random regular syndrome decoding instance (H, y) is *bad*, in a sense formally defined below. The bounds obtained in this section form a core component of the security analysis of the scheme in Section 6.4.2.

6.4.1.1 Bounding the Number of Distinct $\pi(\mathbf{u})\downarrow x$ Solutions

Let $\mathsf{Perm}_w := \mathsf{Perm}(w)$ denote the set of all permutations $\pi : [w] \mapsto [w]$. Given a vector $\mathbf{u} = (\mathbf{u}_1, \cdots, \mathbf{u}_w) \in \mathbb{F}_2^K$, where $(\mathbf{u}_1, \cdots, \mathbf{u}_w)$ forms a decomposition of $\mathbf{u}$ into w blocks $\mathbf{u}_i \in \mathbb{F}_2^{K/w}$, $\pi(\mathbf{u})$ is written to denote the vector $(\mathbf{u}_{\pi(1)}, \cdots, \mathbf{u}_{\pi(w)})$. That is, $\pi(\mathbf{u})$ is the vector obtained by shuffling the w blocks of $\mathbf{u}$ according to the permutation π.
For every $\mathbf{u} \in \mathbb{F}_2^K$, define $\mathsf{pn}(\mathbf{u}) = |\{\pi(\mathbf{u}) \mid \pi \in \mathsf{Perm}([w])\}|$. That is, $\mathsf{pn}(\mathbf{u})$ is the number of distinct vectors in $\mathbb{F}_2^K$ which can be obtained by permuting $\mathbf{u}$ blockwise. Given a bound B, define $\mathsf{PN}_B = \{\mathbf{u} \mid \mathsf{pn}(\mathbf{u}) > B\}$.

Definition 6.4.2 (GOOD_B)**.** *Given a bound B, GOOD_B is defined as the set of syndrome decoding instances $(H, y) \in \mathbb{F}_2^{k \times K} \times \mathbb{F}_2^k$ such that for every $\mathbf{u} \in \mathsf{PN}_B \setminus \mathsf{Reg}_w$ and for all $x^* \in [\mathsf{bs}]^w$,*

$$\Pr_{\pi \leftarrow\!\$ \mathsf{Perm}_w}[H \cdot (\pi(\mathbf{u})\downarrow x^*) = y] \leq \frac{4}{B}.$$

In other words, GOOD_B is the set of syndrome decoding instances (H, y) such that for every $\mathbf{u} \notin \mathsf{Reg}_w$ with at least B distinct blockwise permutations, at most a fraction $4/B$ of all blockwise permutations $\pi(\mathbf{u})$ are close *to being solutions to $H \cdot x = y$, where $\pi(\mathbf{u})$ is said to be "close" to a solution if there exists a suitable cyclic shift of its block $\pi(\mathbf{u})\downarrow x^*$ which is a solution.*

Equipped with this definition, the following lemma is established:

Lemma 6.4.2 (Most syndrome decoding instances are good)**.**

$$\Pr_{H,y}\left[(H,y) \in \mathsf{GOOD}_B\right] > 1 - \varepsilon_{\mathsf{G}},$$

where

$$\varepsilon_{\mathsf{G}} = \binom{2B}{5} \cdot \frac{2^{K+1}}{B \cdot 2^{3k}} \cdot \left(10 + \frac{K^w}{w^w \cdot 2^k}\right).$$

Proof. The proof relies on a small technical lemma, stated below:

Claim 6.4.1. *For any integer $t \leq K$ and every t-tuple of linearly-independent vectors $(\mathbf{v}_1, \cdots, \mathbf{v}_t)$, it holds that*

$$\Pr_{H \leftarrow\$ \mathbb{F}_2^{k \times K}}[H \cdot \mathbf{v}_i = \mathbf{0} \text{ for } i = 1 \text{ to } t] = \frac{1}{2^{k \cdot t}}.$$

Proof. Let V denote the matrix $(\mathbf{v}_1 || \cdots || \mathbf{v}_t)$. Write $V = V^\top // V^\perp$, where $V^\top \in \mathbb{F}_2^{t \times t}$ denotes the invertible square matrix formed by the first t rows of V, and $V^\perp$ denotes the bottom $K - t$ rows. Given a matrix H, write $H = H_L || H_R$, where H_L denotes the t leftmost columns of H, and H_R its remaining columns. It holds that:

$$\begin{aligned}
H \cdot V = 0 &\iff H \cdot [V^\top // V^\perp] = 0 \\
&\iff H \cdot [\mathsf{Id}_t // V^\perp \cdot (V^\top)^{-1}] \cdot V^\top = 0 \\
&\iff (H_L \cdot \mathsf{Id}_t + H_R \cdot V^\perp \cdot (V^\top)^{-1}) \cdot V^\top = 0 \\
&\iff H_R \cdot V^\perp \cdot (V^\top)^{-1} = H_L.
\end{aligned}$$

Therefore, when H is sampled as a uniformly random matrix, $\Pr[H \cdot V = 0] = \Pr[H_R \cdot V^\perp \cdot (V^\top)^{-1} = H_L] = 1/2^{k \cdot t}$, since the right-hand side is a uniformly random matrix $H_L \leftarrow\$ \mathbb{F}_2^{k \times t}$, sampled independently of the left-hand side. The claim follows. □

Now, fix $\mathbf{u} \in \mathsf{PN}_B \setminus \mathsf{Reg}_w$ and $x^* \in [\mathsf{bs}]^w$. Let $N \leftarrow \mathsf{pn}(\mathbf{u})$ and $\mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(N)}$ be the lexical ordering of all distinct vectors of the form $\pi(\mathbf{u})$ for some $\pi \in \mathsf{Perm}_w$. Fix any subset $S = i_1, \cdots, i_5 \subset [N]$ of five indices. In the following, the probability

$$p(S) = \Pr_{H,y}[H \cdot (\mathbf{u}^{(i_1)} \downarrow x^*) = y \wedge \cdots \wedge H \cdot (\mathbf{u}^{(i_5)} \downarrow x^*) = y]$$

will be bounded. Recall that a regular syndrome decoding instance (H, y) is sampled by picking a uniformly random matrix $H \leftarrow\$ F_2^{k \times K}$, a uniformly random regular vector $x \leftarrow\$ \mathsf{Reg}_w$, and setting $y \leftarrow H \cdot x$. When making the sampling of x explicit, the probability

$p(S)$ rewrites to

$$p(S) = \Pr_{H,x}[H \cdot (\mathbf{u}^{(i_1)} \downarrow x^* \oplus x) = \mathbf{0} \wedge \cdots \wedge H \cdot (\mathbf{u}^{(i_5)} \downarrow x^* \oplus x) = \mathbf{0}].$$

Now, write $(\mathbf{v}_1, \cdots, \mathbf{v}_5) \leftarrow (\mathbf{u}^{(i_1)} \downarrow x^* \oplus x, \cdots, \mathbf{u}^{(i_5)} \downarrow x^* \oplus x)$, which are random variables defined over the sampling of x, and let Z_S denote the event (defined over the sampling of both x and H) that $H \cdot \mathbf{v}_i = \mathbf{0}$ for $i = 1$ to 5 (in other words, $p(S) = \Pr[Z_S]$). If the vectors $(\mathbf{v}_1, \cdots, \mathbf{v}_5)$ were guaranteed to be linearly independent, it would immediately follow that $p(S) = \Pr[Z_S] = 1/2^{5k}$ by the previous claim; however, they are not necessarily independent, and a more fine-grained approach is required. To bound $p(S)$, a few simple observations are made:

- Since $\mathbf{u} \notin \mathsf{Reg}_w$, it also holds that for any permutation π and shifts x^*, $\pi(\mathbf{u}) \downarrow x^* \notin \mathsf{Reg}_w$ (since shuffling the blocks and cyclically shifting each block yields an invertible mapping that preserves regularity). This implies that $\mathbf{v}_j \neq \mathbf{0}$ holds with probability 1 for $j = 1$ to 5 (since $\mathbf{v}_j = 0 \iff \mathbf{u}^{(i_1)} \downarrow x^* = x$, and $x \in \mathsf{Reg}_w$).
- Because the $\mathbf{u}^{(i)}$ are pairwise distinct (by definition), the $\mathbf{v}_j$ are pairwise distinct.

Equipped with these observations, let denote E_S the event that there exist three integers $\alpha \neq \beta \neq \gamma \in [5]$ such that $\mathbf{v}_\alpha \oplus \mathbf{v}_\beta \oplus \mathbf{v}_\gamma = 0$. Observe that

$$\begin{aligned}
\Pr[E_S] &= \Pr_x[\exists \alpha \neq \beta \neq \gamma \in [5] \ : \ (\mathbf{u}^{(i_\alpha)} \downarrow x^* \oplus x) \oplus (\mathbf{u}^{(i_\beta)} \downarrow x^* \oplus x) \oplus (\mathbf{u}^{(i_\gamma)} \downarrow x^* \oplus x) = 0] \\
&= \Pr_x[\exists \alpha \neq \beta \neq \gamma \in [5] \ : \ (\mathbf{u}^{(i_\alpha)} \downarrow x^*) \oplus (\mathbf{u}^{(i_\beta)} \downarrow x^*) \oplus (\mathbf{u}^{(i_\gamma)} \downarrow x^*) = x] \\
&\leq \sum_{\alpha \neq \beta \neq \gamma} \Pr_x[(\mathbf{u}^{(i_\alpha)} \downarrow x^*) \oplus (\mathbf{u}^{(i_\beta)} \downarrow x^*) \oplus (\mathbf{u}^{(i_\gamma)} \downarrow x^*) = x] \\
&\leq \binom{5}{3} \cdot \left(\frac{K}{w}\right)^{-w} = 10 \cdot \left(\frac{K}{w}\right)^{-w},
\end{aligned}$$

which follows from a union bound over all possible size-3 subsets of $[5]$ and because there are $(K/w)^w$ vectors in Reg_w, hence a $(K/w)^{-w}$ probability (at most) that a random vector $x \leftarrow\!\!\$\ \mathsf{Reg}_w$ is equal to the fixed vector $(\mathbf{u}^{(i_\alpha)} \downarrow x^*) \oplus (\mathbf{u}^{(i_\beta)} \downarrow x^*) \oplus (\mathbf{u}^{(i_\gamma)} \downarrow x^*)$. Now, it holds that

$$\begin{aligned}
\Pr[Z_S] &= \Pr[Z_S \mid E_S] \cdot \Pr[E_S] + \Pr[Z_S \mid \neg E_S] \cdot \Pr[\neg E_S] \\
&\leq 10 \cdot (K/w)^{-w} \cdot \Pr[Z \mid E_S] + \Pr[Z \mid \neg E_S].
\end{aligned}$$

Now, bound $\Pr[Z_S; |; E_S]$. For simplicity and without loss of generality, assume that after sampling x, $\mathbf{v}_1 \oplus \mathbf{v}_2 \oplus \mathbf{v}_3 = 0$ (this is without loss of generality because the $\mathbf{v}_i$'s can always be reordered after sampling x; note that the event E_S is defined only over the sampling of x). Then, because $\mathbf{v}_4 \neq \mathbf{v}_3$, it necessarily holds that $\mathbf{v}_1 \oplus \mathbf{v}_2 \oplus \mathbf{v}_4 \neq 0$. Furthermore, since

the $\mathbf{v}_i$ are all pairwise distinct, and all nonzero, this implies that $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_4)$ are linearly independent. Then, using the claim:

$$\Pr[Z_S \mid E_S] \leq \Pr[H \cdot \mathbf{v}_1 = 0 \wedge H \cdot \mathbf{v}_2 = 0 \wedge H \cdot \mathbf{v}_4 = 0 \mid E_S] = \frac{1}{2^{3k}}.$$

Next, bound $\Pr[Z_S; |; \neg E_S]$. By similar reasoning, after sampling x, it necessarily holds that there is a 4-tuple of the $\mathbf{v}_i$'s that does not XOR to 0 (since if all 4-tuples of the $\mathbf{v}_i$'s XOR to 0, then $\mathbf{v}_1 \oplus \mathbf{v}_2 \oplus \mathbf{v}_3 \oplus \mathbf{v}_4 = \mathbf{v}_1 \oplus \mathbf{v}_2 \oplus \mathbf{v}_3 \oplus \mathbf{v}_5 = 0$, which implies $\mathbf{v}_4 = \mathbf{v}_5$, contradicting the fact that the $\mathbf{v}_i$'s are pairwise distinct). Without loss of generality, assume that $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4)$ do not XOR to 0. Because the condition $\neg E_S$ is applied, it also holds that no 3-tuple of vectors from $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4)$ XOR to 0, and because the $\mathbf{v}_i$'s are pairwise distinct (*i.e.*, no two-tuple XOR to 0) and nonzero, it follows that $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4)$ are linearly independent. By the previous claim:

$$\Pr[Z_S \mid \neg E_S] \leq \Pr[H \cdot \mathbf{v}_1 = 0 \wedge H \cdot \mathbf{v}_2 = 0 \wedge H \cdot \mathbf{v}_3 = 0 \wedge H \cdot \mathbf{v}_4 = 0 \mid E] = \frac{1}{2^{4k}}.$$

Eventually, it follows that:

$$\begin{aligned} p(S) = \Pr[Z_S] &\leq 10 \cdot (K/w)^{-w} \cdot \Pr[Z_S \mid E_S] + \Pr[Z_S \mid \neg E_S] \\ &\leq \frac{1}{2^{3k}} \cdot \left(10 \cdot \left(\frac{K}{w}\right)^{-w} + \frac{1}{2^k}\right). \end{aligned}$$

To complete the proof of Lemma 6.4.2, a careful union bound is used. Given $\mathbf{u} \in \mathsf{PN}_B \setminus \mathsf{Reg}_w$ and $x^* \in [\mathsf{bs}]^w$, partition the $N = \mathsf{pn}(\mathbf{u})$ vectors $\mathbf{u}^{(i)} \downarrow x^*$ into $m \leq N/B$ blocks of at most $2B$ vectors each. Let $N_1, \cdots, N_m$ denote the m disjoint subsets $N_i \subset [N]$ of size $|N_i| \leq B$ corresponding to this partition. First, apply a union bound over all possible blocks N_i, and all possible size-5 subsets of N_i:

$$\begin{aligned} &\Pr_{H,x}[\exists i \leq m, \exists S_i \subset N_i \subset [N] \text{ with } |S_i| = 5 \ : \ H \cdot (\mathbf{u}^{(j)} \downarrow x^* \oplus x) = \mathbf{0} \text{ for all } j \in S_i] \\ &\leq m \cdot \binom{2B}{5} \cdot \frac{1}{2^{3k}} \cdot \left(10 \cdot \left(\frac{K}{w}\right)^{-w} + \frac{1}{2^k}\right). \end{aligned}$$

This implies that for any fixed $\mathbf{u} \in \mathbb{F}_2^K$ with $\mathsf{pn}(\mathbf{u}) = N$, and any fixed $x^{\in}[\mathsf{bs}]^w$, there are at most $4 \cdot m$ indices $j \in [N]$ such that $H \cdot (\mathbf{u}^{(j)} \downarrow x^{\oplus} x) = \mathbf{0}$ with high probability (since with high probability, in each of the m blocks, there are at most 4 such indices):

$$
\begin{aligned}
&1 - m \cdot \binom{2B}{5} \cdot \frac{1}{2^{3k}} \cdot \left(10 \cdot \left(\frac{K}{w}\right)^{-w} + \frac{1}{2^k}\right) \\
&< \Pr_{H,x}[\forall i \leq m, \forall S_i \subset N_i \subset [N] \text{ with } |S_i| = 5 \ : \ \exists j \in S_i, H \cdot (\mathbf{u}^{(j)} \downarrow x^* \oplus x) = \mathbf{0}] \\
&= \Pr_{H,x}[\forall i \leq m \ : \ \text{ there are at most 4 } j \in N_i \text{ s.t. } H \cdot (\mathbf{u}^{(j)} \downarrow x^* \oplus x) = \mathbf{0}] \\
&\leq \Pr_{H,x}[\exists \leq 4 \cdot m \text{ indices } j \in [N] \text{ such that } H \cdot (\mathbf{u}^{(j)} \downarrow x^* \oplus x) = \mathbf{0} \text{ for all } j \in S_i] \\
&= \Pr_{H,x}\left[\Pr_{\pi \in \mathsf{Perm}_w}[H \cdot (\pi(\mathbf{u}) \downarrow x^* \oplus x) \neq \mathbf{0}] \leq \frac{4 \cdot N/B}{N} = \frac{4}{B}\right].
\end{aligned}
$$

Next, a union bound is computed over all possible vectors $\mathbf{u}$ with permutation number $\mathsf{pn}(\mathbf{u}) = N$ (where $N \leq w!$, with equality when all blocks of $\mathbf{u}$ are distinct) and all shifts $x^* \in [\mathsf{bs}]^w$. For any $N \in [w!]$, let $n(N)$ denote the total number of vectors $\mathbf{u} \in \mathbb{F}_2^K$ with $\mathsf{pn}(\mathbf{u}) = N$ (note that $\sum_{i \in [w!]} n(N) = 2^K$). All vectors $\mathbf{u}$ with $\mathsf{pn}(\mathbf{u}) = N$ are grouped into $n(N)/N$ equivalence classes $U_1, \cdots, U_{n(N)/N}$, where two vectors $\mathbf{u}_1, \mathbf{u}_2$ belong to the same equivalence class U_i if and only if there exists $\pi \in \mathsf{Perm}_w$ such that $\mathbf{u}_1 = \pi(\mathbf{u}_2)$ (note that each equivalence class is of size exactly N by definition of pn, and the U_i form a partition of the set $\mathbf{u} \in \mathbb{F}_2^K ; : ; \mathsf{pn}(\mathbf{u}) = N$). An important observation is that, because any two vectors $\mathbf{u}_1, \mathbf{u}_2$ that belong to the same equivalence class U_i generate the exact same N-tuple of distinct permuted vectors $(\mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(N)})$ (ordered lexically), it suffices to perform the union bound over all possible *equivalence classes* $(U_1, \cdots, U_{n(N)/N})$, and over all shifts x^*:

$$
\begin{aligned}
&\Pr_{H,x}\left[\exists i \leq n(N)/N, \exists x^* \in [\mathsf{bs}]^w, \Pr_{\pi \in \mathsf{Perm}_w}[H \cdot (\pi(\mathbf{u}) \downarrow x^* \oplus x) = \mathbf{0}] > \frac{4}{B}\right] \\
&\leq \frac{n(N)}{N} \cdot \frac{N}{B} \cdot \binom{2B}{5} \cdot \left(\frac{K}{w}\right)^w \cdot \frac{1}{2^{3k}} \cdot \left(10 \cdot \left(\frac{K}{w}\right)^{-w} + \frac{1}{2^k}\right),
\end{aligned}
$$

where the vector $\mathbf{u}$ in the probability denote any representent of the class U_i. Eventually, using a union bound over all possible values $N \in [w!]$ with $N \geq B$:

$$
\begin{aligned}
&\Pr_{H,y}[(H, y) \notin \mathsf{GOOD}_B] \\
&= \Pr_{H,x}\left[\exists N \geq B \in [w!], \exists i \leq n(N)/N, \exists x^* \in [\mathsf{bs}]^w, \Pr_{\pi \in \mathsf{Perm}_w}[H \cdot (\pi(\mathbf{u}) \downarrow x^* \oplus x) = \mathbf{0}] > \frac{4}{B}\right] \\
&\leq \binom{2B}{5} \cdot \left(\frac{K}{w}\right)^w \cdot \frac{1}{B \cdot 2^{3k}} \cdot \left(10 \cdot \left(\frac{K}{w}\right)^{-w} + \frac{1}{2^k}\right) \cdot \sum_{N=B}^{w!} n(N) \\
&< \binom{2B}{5} \cdot \frac{2^{K+1}}{B \cdot 2^{3k}} \cdot \left(10 + \frac{K^w}{w^w \cdot 2^k}\right),
\end{aligned}
$$

which concludes the proof of Lemma 6.4.2. □ □

6.4.2 Security Analysis

In this section, Theorem 6.3.1 is proven.

6.4.2.1 Reducing to EUFKO Security

The following lemma is now proven:

Lemma 6.4.3 (EUFKO $\implies$ EUF-CMA)**.**

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{EUFKO}}_{\mathcal{A}} + \frac{q_s\,(q_s + q_1 + q_2 + q_3)}{2^{2\lambda}} + \epsilon_{\mathsf{F}} + \epsilon_{\mathsf{PRG}}$$

Proof. Consider an adversary $\mathcal{A}$ against the EUF-CMA property of the signature scheme. To prove security, a sequence of experiments involving $\mathcal{A}$ is defined, where the first corresponds to the experiment in which $\mathcal{A}$ interacts with the real signature scheme, and the last one is an experiment in which $\mathcal{A}$ uses only random elements independent of the witness.

Game 1 (Gm^1). This corresponds to the actual interaction of $\mathcal{A}$ with the real signature scheme. The probability of the event Forge, i.e., the event that $\mathcal{A}$ can generate a valid signature for a message that was not previously queried to the signing oracle, needs to be bounded.

Game 2 (Gm^2). In this step, the game aborts if the sampled salt salt collides with the value sampled in any of the previous queries to hash functions H_1 or H_2, or if the input of PRG_2 collides with the value obtained in any of the previous queries. The probability difference can be bounded as

$$|\Pr[\mathsf{Gm}^1(\mathsf{Forge})] - \Pr[\mathsf{Gm}^2(\mathsf{Forge})]| \leq \frac{q_s \cdot (q_s + q_1 + q_2 + q_3)}{2^{2\lambda}}$$

Game 3 (Gm^3). The difference from the previous game is that, before signing a message, uniformly random values h_1, h_2, and i^* are chosen. Since Phase 1, Phase 3, and Phase 5 are computed as before, and the only change is setting the output of H_1 as h_1, H_2 as h_2, and $\mathsf{PRG}_2(h_2)$ as i^*, the difference in forgery probability arises only if a query to H_1, H_2, or PRG_2 was ever made before, but in this scenario, Game 2 aborts. Thus:

$$\Pr[\mathsf{Gm}^2(\mathsf{Forge})] = \Pr[\mathsf{Gm}^3(\mathsf{Forge})]$$

Game 4 (Gm^4) In this game, the i^*-th seed sd_{i^*} and the related co-path CoPath_{i^*} are sampled at random. Using all the seeds $\mathsf{sd}_{i i \neq i^*}$ in the CoPath_{i^*}, all the parties' views and the auxiliary material are computed. Phase 1 and Phase 3 are executed in the actual way (i.e., using the real witness), except for i^*, for which values are obtained randomly instead of using the F. Distinguishing between this game and the previous one is equivalent to breaking the multi-instance security of the PPRF:

$$|\Pr[\mathsf{Gm}^4(\mathsf{Forge})] - \Pr[\mathsf{Gm}^6(\mathsf{Forge})]| \leq \epsilon_{\mathsf{F}}$$

Game 5 (Gm^5). Before signing a message, a uniformly random value is chosen to be used as the i^*-th party's view, i.e., $(x_{i^*}, r_{i^*}, u_{i^*})$, and its commitment com_{i^*}. Since in the previous game, these values were computed using a multi-instance PRG on a random sd, with salt salt, the following bound holds:

$$|\Pr[\mathsf{Gm}^4(\mathsf{Forge})] - \Pr[\mathsf{Gm}^6(\mathsf{Forge})]| \leq \epsilon_{\mathsf{PRG}}$$

Game 6 (Gm^6) In this game, Phase 1 and Phase 3 are changed by having the signer use the internal HVZK simulator described in Protocol below 16.

HVZK simulator

Step 1: (Sample Challenge).

1. Sample $\mathsf{CH}_1 = \pi \in \mathsf{Perm}([w])$ and $\mathsf{CH}_2 = i^* \in [n]$ and $\mathsf{salt} \in \{0,1\}^{2\lambda}$.

Step 2: (Sample Leaf Party States).

1. Sample $(x_{i^*}, r_{i^*}, u_{i^*}, \mathsf{com}_{i^*}) \leftarrow\!\!\$\ [\mathsf{bs}]^w \times [\mathsf{bs}]^w \times \{0,1\}^K \times \{0,1\}^\lambda$;
2. Sample the CoPath_{i^*} at random;
3. Sample $\mathsf{aux} \leftarrow\!\!\$\ [\mathsf{bs}]^w \times \{0,1\}^K$.

Step 3: (Generate Leaf Party Commitments).

1. For $i \neq i^*$:
   - If $i \neq n$:
     - Expand the leaf party into shares (x_i, r_i, u_i) and commitment com_i by using a PRG on sd_i;
   - If $i = n$:
     - Set $\mathsf{state}_n = \mathsf{sd}_n || \mathsf{aux}$ and compute $\mathsf{com}_n = H(\mathsf{state}_n)$;
     - Recompute r_n s.t. $u = \mathsf{Expand}(r)$ where $u = \sum_{i=1}^n u_i$ and $r = \sum_{i=1}^n r_i$.
2. Compute $\mathsf{COM} = H_1(m, \mathsf{salt}, \mathsf{com}_1, \ldots, \mathsf{com}_n)$.

Step 4: (Generate party communication).

1. Sample $z \in [\mathsf{bs}]^w$ at random;
2. For $d = 1$ to D:
   - Set $(X_{d,0}, R_{d,0}, U_{d,0}) \leftarrow (0,0,0) \in [\mathsf{bs}]^w \times [\mathsf{bs}]^w \times \{0,1\}^K$;
   - Compute
     - $X_{d,0} \leftarrow X_{d,0} + x_i \bmod \mathsf{bs}$;

- $R_{d,0} \leftarrow R_{d,0} + r_i \bmod \mathsf{bs}$;
- $U_{d,0} \leftarrow U_{d,0} \oplus u_i$;
- $y_{d,0} \leftarrow H \cdot \mathsf{Shift}(\pi(U_{d,0}), z)$;
- $y_{d,1} \leftarrow y_{d,0} \oplus y$;
- $z_{d,0} \leftarrow X_d - \pi(R_{d,0}) \mod \mathsf{bs}$;
- $z_{d,1} \leftarrow z_{d,0} - z \bmod \mathsf{bs}$.

Step 5: (Output transcript).

1. $\mathsf{RSP}_1 = \mathsf{H}_2\left(m, \mathsf{salt}, \mathsf{COM}, (y_{d,b}, z_{d,b})_{d \le D, b \in \{0,1\}}\right)$;
2. Program PRG_2 as a ROM s.t. $\mathsf{PRG}_2(\mathsf{RSP}_1) = \mathsf{CH}_2$;
3. $\mathsf{RSP}_2 = \mathsf{com}_{i^*}, \mathsf{CoPath}_{i^*}, \mathsf{aux}_n$.

Output $(\mathsf{COM}, \mathsf{RSP}_1, \mathsf{RSP}_2)$.

Protocol 16: Internal HVZK simulator for signing algorithm

Looking in detail, the only change between this game and the previous one is that the auxiliary material aux is now selected as random. Since in the previous game, aux was computed using all real values except one (randomly chosen and never made public), there is no substantial difference between this game and the previous one. Therefore,

$$\Pr[\mathsf{Gm}^5(\mathsf{Forge})] = \Pr[\mathsf{Gm}^6(\mathsf{Forge})]$$

Game 7 (Gm^7). An execution e^* of a query

$$h_2 = \mathsf{H}_2(m, \mathsf{salt}, h_1, (y^e_{d,b}, z^e_{d,b})_{d \le D, b \in \{0,1\}, e \le \tau})$$

is said to define a correct witness if the following criteria are satisfied:

- h_1 was output by a previous query

$$h_1 \leftarrow \mathsf{H}_1(m, \mathsf{salt}, \mathsf{com}^1_1, \cdots, \mathsf{com}^1_N, \cdots, \mathsf{com}^\tau_1, \cdots, \mathsf{com}^\tau_N);$$

- each $\mathsf{com}^{e^*}_i$ in this query was output by a previous query

$$\mathsf{com}^{e^*}_i = \mathsf{PRG}(\mathsf{sd}^{e^*}, i)$$

  for each $i \in [N]$;

- The vector x defined by the leaf party states $\{\mathsf{state}_i\}_{i \in N^D}$ satisfies $\mathsf{HW}(x) = w$ and $Hx = y$.

In this game, for each query of H_2 made by the adversary, a check is performed to determine if there is an execution e^* that defines a correct witness. Calling this event Solve of course, since if it occurs then the states $\{\mathsf{state}_i^{e^*}\}$ define a solution for the RSD, the probability $\Pr[\mathsf{Solve} \leq \epsilon_{\mathsf{SD}}]$.

□

6.4.2.2 EUFKO Security

The following lemma is proven:

Lemma 6.4.4 (EUFKO security)**.**

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{EUFKO}} \leq \varepsilon_{\mathsf{SD}} + \Pr[X + Y = \tau] + \varepsilon_{\mathsf{G}} + \frac{1}{2^{\lambda}}.$$

Together with Lemma 6.4.3, this completes the proof of Theorem 6.3.1. To prove EUFKO security of the signature scheme, the soundness of the underlying identification scheme is first analyzed, and then the standard reduction to EUFKO security is applied after compiling the scheme with the Fiat-Shamir transform. Concretely, the signature scheme is obtained by applying the Fiat-Shamir transform to the τ-fold parallel repetition of the identification scheme defined in Protocol 17.

Five-round Identification Scheme

1. Parse the secret key sk as (sd, x);
2. Let $H \leftarrow \mathsf{PRG}(\mathsf{sd})$ and $y \leftarrow H \cdot \mathsf{Expand}(x)$;
3. Sample $(K_0, K_1) \leftarrow\!\!\$\ \{0,1\}^{\lambda} \times \{0,1\}^{\lambda}$. Set $\mathsf{salt} \leftarrow (K_0, K_1)$.

Round 1: (Prover to Verifier).

1. Sample $\mathsf{sd} \leftarrow\!\!\$\ \{0,1\}^{\lambda}$;
2. For $d = 1$ to D:
   - Set $(X_{d,0}, R_{d,0}, U_{d,0}) \leftarrow (0,0,0) \in [\mathsf{bs}]^w \times [\mathsf{bs}]^w \times \{0,1\}^K$;
   - Set $x_n \leftarrow x$, $u_n \leftarrow 0$, and $r \leftarrow 0$.
3. **For** $i = 1$ **to** $n-1$:
   - Compute $\mathsf{sd}_i \leftarrow \mathsf{F}_{\mathsf{salt}}(\mathsf{sd}, i)$;
     Set $\mathsf{state}_i \leftarrow \mathsf{sd}_i$ and $(x_i, r_i, u_i, \mathsf{com}_i) \leftarrow \mathsf{PRG}(\mathsf{sd}_i)$;
   - Set $x_n \leftarrow x_n - x_i \bmod \mathsf{bs}$, $u_n^e \leftarrow u_n \oplus u_i$, and $r \leftarrow r + r_i \bmod \mathsf{bs}$;
   - For all $d \leq D$ such that $i[d] = 0$:
     Set $X_{d,0} \leftarrow X_{d,0} + x_i \bmod \mathsf{bs}$, $R_{d,0} \leftarrow R_{d,0} + r_i \bmod \mathsf{bs}$,
     and $U_{d,0} \leftarrow U_{d,0} \oplus u_i$.

4. **On node** n**:**
   - Compute $\mathsf{sd}_n \leftarrow \mathsf{F}_{\mathsf{salt}}(\mathsf{sd}, n)$, $r_n \leftarrow \mathsf{PRG}(\mathsf{sd}_n)$, and set $r \leftarrow r + r_n \bmod \mathsf{bs}$, $u \leftarrow \mathsf{Expand}(r)$, and $u_n \leftarrow u_n \oplus u$;
   - Define $\mathsf{aux}_n \leftarrow (x_n, u_n)$ and set $\mathsf{state}_n \leftarrow \mathsf{aux}_n || \mathsf{sd}_n$ and $\mathsf{com}_n \leftarrow \mathsf{H}(\mathsf{state}_n)$;
   - Compute and send $h_1 \leftarrow \mathsf{H}_1(\mathsf{salt}, \mathsf{com}_1, \cdots, \mathsf{com}_N)$.

Round 2: (Verifier to Prover).

1. Send $\pi \leftarrow\!\!\$\ \mathsf{Perm}([w])$.

Round 3: (Prover to Verifier).

1. Set $z \leftarrow x - \pi(r) \bmod \mathsf{bs}$;
2. For $d = 1$ to D:
   - Set $y_{d,0} \leftarrow H \cdot \mathsf{Shift}(\pi(U_{d,0}), z)$, $y_{d,1} \leftarrow y_{d,0} \oplus y$, $z_{d,0} \leftarrow X_d - \pi(R_{d,0}) \bmod \mathsf{bs}$, and $z_{d,1} \leftarrow z_{d,0} - z \bmod \mathsf{bs}$;
   - Compute and send $h_2 \leftarrow \mathsf{H}_2(\mathsf{salt}, h_1, (y_{d,b}, z_{d,b})_{d \le D, b \in \{0,1\}})$.

Round 4: (Verifier to prover).

1. Send $(b_1, \cdots b_D) \leftarrow\!\!\$\ \{0,1\}^D$. Let $i \leftarrow \sum_{d=1}^{D} b_d \cdot 2^{d-1}$.

Round 5: (Prover to Verifier).

1. Send $(\mathsf{salt}, z, (\mathsf{CoPath}_{\mathsf{salt}}(i, \mathsf{sd}), \mathsf{com}_i, \mathsf{aux}_n))$.

Protocol 17: A five-round identification scheme with secret key $\mathsf{sk} = (\mathsf{sd}, x)$ *for the relation* $y = H \cdot x$ *with* $y \in \mathsf{Reg}_w$ *and* $H = \mathsf{PRG}(\mathsf{sd})$*. The scheme has soundness* $\varepsilon = \mathsf{p} + (1 - \mathsf{p})/n$.

6.4.2.3 Soundness of the identification scheme. The core of the analysis is dedicated to showing that from any cheating prover, a *weakly valid witness* can be extracted. Concretely, a weakly valid witness is a pair $(\mathbf{v}, x^*)$ where $\mathbf{v}$ is a solution to $H \cdot \mathbf{v} = y$ which might not be regular, but satisfies $\mathsf{pn}(\mathbf{v}{\downarrow}x^*) \le B$. In other words, this means that $\mathbf{v}$ contains mostly identical blocks "up to shift". Note that a regular vector contains only copies of the unit vector $\mathbf{e}_1$ "up to shift", hence this generalizes the class of regular vectors in a specific sense. Formally:

Definition 6.4.3. *A* weakly valid witness *to a syndrome decoding instance* (H, y) *is a pair* $(\mathbf{v}, x^*)$ *such that* $H \cdot \mathbf{v} = y$ *and* $\mathbf{v} \in X$*, where* X *is defined as:*

$$X = \{\mathbf{v} \in \mathbb{F}_2^K \ : \ \exists \mathbf{u} \in \mathbb{F}_2^K \setminus \mathsf{PN}_B, \exists x^* \in [\mathsf{bs}]^w, \mathbf{v} = \mathbf{u}{\downarrow}x^*\}.$$

Additionally, the second term x^* *of the pair is a shift that satisfies* $\mathbf{v}{\downarrow}x^* \in \mathbb{F}_2^K \setminus \mathsf{PN}_B$.

Lemma 6.4.5 (Soundness of the identification scheme in Protocol 17)**.** *Assume that* $\mathsf{H}_1, \mathsf{H}_2$ *are collision-resistant hash functions, that the mapping* $\mathsf{PRG}(\mathsf{sd})_{1..\lambda}$ *(*i.e. *the first* λ *bits of the output of* PRG *on an input* sd*) is computationally binding, and that the PRG used during the key generation in* $H \leftarrow \mathsf{PRG}(\mathsf{sd})$ *is modeled as a random oracle (hence* $\mathsf{PRG}(\mathsf{sd})$ *selects a truly random matrix* H*). Then with probability at least* $1 - 1/2^{\lambda} - \varepsilon_{\mathsf{G}}$ *over the random choice of the RSD instance* (H, y)*, there exists an expected polynomial time extractor algorithm which, given rewinding access to a prover* $\tilde{\mathsf{P}}$ *which generates an accepting proof with probability at least* $\tilde{\varepsilon} > \mathsf{p} + 1/n - \mathsf{p}/n$*, extracts a* weakly valid witness x *for the relation* $H \cdot x = y$*.*

Looking ahead, the soundness proof does not prevent a cheating prover from producing a weakly valid witness that is not a true regular witness. This is guaranteed by the fact that, with probability $1 - 1/2^{\lambda}$ over the choice of a random instance (H, y), the system of equations $H \cdot \mathbf{v} = y$ does not have *any* solution in X beyond the regular solution. This holds for a suitable choice of the parameters (K, k, w), which is detailed in Section 6.3.2. The term $1/2^{\lambda}$ in the bound of Lemma 6.4.4 reflects the probability that (H, y) admits weakly valid solutions that are not regular.

Proof. Let $\tilde{\mathsf{P}}$ be a prover that succeeds in generating an accepting proof with probability $\tilde{\varepsilon} > \varepsilon$. An *extractor* is exhibited, which finds a witness x such that $H \cdot x = y$, where x is guaranteed to be a *weakly valid* witness (see Definition 6.4.3). Let R denote the randomness used by $\tilde{\mathsf{P}}$ to generate the commitment h in the first round, and by R^* a possible realization of R. Let $\mathsf{Succ}_{\tilde{\mathsf{P}}}$ denote the event that $\tilde{\mathsf{P}}$ succeeds in convincing V. By hypothesis

$$\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}] = \tilde{\varepsilon} > \varepsilon = \mathsf{p} + \frac{1}{N} - \frac{\mathsf{p}}{N}.$$

Fix an arbitrary value $\alpha \in \{0, 1\}$ such that $(1 - \alpha)\tilde{\varepsilon} > \varepsilon$, which exists since $\tilde{\varepsilon} > \varepsilon$. A realization R^* of the prover randomness for the first flow is said to be *good* if it holds that

$$\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}} | R = R^*] \geq (1 - \alpha)\tilde{\varepsilon}.$$

Furthermore, by the Splitting Lemma (see e.g. [FJR22]), it follows that $\Pr[R \text{ good} | \mathsf{Succ}_{\tilde{\mathsf{P}}}] \geq \alpha$. Assume now that T_0 is the transcript of a successful execution of the zero-knowledge proof with $\tilde{\mathsf{P}}$. Let R^* denote the random coin used by $\tilde{\mathsf{P}}$ in the first round, and let i_0 denote the Round 4 message of the verifier. If R^* is *good*, then

$$\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}} | R = R^*] \geq (1 - \alpha)\tilde{\varepsilon} > \varepsilon > \frac{1}{N},$$

which implies that there necessarily exists a second successful transcript T_1 with a different Round 4 message $i_1 \neq i_0$.

Consistency of (T_0, T_1)**.** Let (π_0, i_0) and (π_1, i_1) be the verifier challenges in the successful transcripts T_0 and T_1 respectively, with $i_0 \neq i_1$. Let $(\mathsf{state}^0_{i \neq i_0}, \mathsf{com}^0_{i_0})$ and $\left(\mathsf{state}^1_{i \neq i_1}, \mathsf{com}^1_{i_1}\right)$ denote the states (recomputed from the co-path included in the transcript) and the commit-

ment in the transcripts T_0 and T_1 respectively. Suppose that $\exists i \in [N] \setminus \{i_0, i_1\}$ such that $\mathsf{state}_i^0 \neq \mathsf{state}_i^1$. Then there are two possibilities:

- The commitments are different:

$$\mathsf{com}_i = \mathsf{PRG}(\mathsf{state}_i)_{1..\lambda} \neq \mathsf{PRG}(\mathsf{state}'_i)_{1..\lambda} = \mathsf{com}'_i.$$

  But since T_0 and T_1 are accepting transcripts, this implies in particular that $h = \mathsf{H}_1(\mathsf{com}_1, \cdots, \mathsf{com}_N)$ and $h = \mathsf{H}_1(\mathsf{com}'_1, \cdots, \mathsf{com}'_N)$ which contradicts the collision resistance of H_1.

- The commitments are equal:

$$\mathsf{com}_i = \mathsf{PRG}(\mathsf{state}_i)_{1..\lambda} = \mathsf{PRG}(\mathsf{state}'_i)_{1..\lambda} = \mathsf{com}'_i.$$

  This directly contradicts the binding property of PRG.

Therefore, it necessarily holds that the states are mutually consistent (that is $\mathsf{state}^0_{i\neq i_0,i_1} = \mathsf{state}^1_{i\neq i_0,i_1}$. Since $i_0 \neq i_1$), they jointly define a unique tuple $(\mathsf{state}_i)_{i\in[N]}$, from which it is possible to recompute $x = \sum_i x_i \bmod \mathsf{bs}$, $u = \bigoplus_i u_i$, and $r = \sum_i r_i \bmod \mathsf{bs}$. Let denote $v \leftarrow u \uparrow r$.

Claim 6.4.2. *The vector v belongs to $\mathbb{F}_2^K \setminus \mathsf{PN}_B$.*

To prove the claim, it is shown that if $v \in \mathsf{PN}_B$, then $\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R = R^*] \leq \varepsilon$, contradicting the assumption that R^* is good. Let $\mathsf{BadPerm} = \mathsf{BadPerm}_{v,x}$ denote the event (defined over the random choice of a permutation π, and for the fixed value of (v, x, H, y)) that $y = H \cdot (\pi(v) \downarrow x)$. Let ε_{G} denote the bound of Lemma 6.4.2.
By Lemma 6.4.2, it holds with probability $1 - \varepsilon_{\mathsf{G}}$ over the random choice of H that $\Pr[\mathsf{BadPerm}] \leq \mathsf{p}$ with $\mathsf{p} = 4/B$ (here, the fact that in the random oracle model, $H = \mathsf{PRG}(\mathsf{sd})$ is uniformly random is used). Now,

$$\begin{aligned}\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R = R^*] &= \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}} \wedge \mathsf{BadPerm}|R = R^*] + \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}} \wedge \neg\mathsf{BadPerm}|R = R^*] \\ &\leq \mathsf{p} + (1 - \mathsf{p}) \cdot \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R = R^* \wedge \neg\mathsf{BadPerm}].\end{aligned}$$

The probability $\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R = R^* \wedge \neg\mathsf{BadPerm}]$ is now bounded. Assume for the sake of contradiction that $\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R = R^* \wedge \neg\mathsf{BadPerm}] > 1/n$. This implies that given any successful transcript T'_0 with fourth-round i'_0, there necessarily exists a second successful transcript T'_1 *with the same first three rounds* and a different fourth-round $i'_1 \neq i'_0$. Fix two such transcripts (T'_0, T'_1), and let π' denote the (common) permutation sent in Round 2 of these transcripts.
By the same argument as before, T'_0 and T'_1 are necessarily consistent, and uniquely define a tuple $(\mathsf{state}'_i)_{i\in[N]}$. Furthermore, since the condition on $R = R^*$ implies that the first flow

h'_1 is the same as the first flow h_1 in T_0, T_1, it must holds that $(\mathsf{state}'_i)_{i\in[N]} = (\mathsf{state}_i)_{i\in[N]}$, the states uniquely defined by (T_0, T_1) (otherwise, this would contradict either either the collision-resistance of H or the binding of PRG, as already shown).
Let $d \leq D$ be a position such that $i'_0[d] \neq i'_1[d]$. Without loss of generality (since the roles of T'_0 and T'_1 can always be swapped), it is safe to assume that $i'_0[d] = 0$ and $i'_1[d] = 1$. Reconstruct the values $(y^{(0)}_{b,d}, z^{(0)}_{b,d})_{b\in\{0,1\}}$ using the seeds $(\mathsf{sd}_i)_{i\neq i'_0}$ and the permutation π' from transcript T'_0, using the same procedure as the verification procedure. Similarly, reconstruct the values $(y^{(1)}_{b,d}, z^{(1)}_{b,d})_{d\leq D, b\in\{0,1\}}$ using the seeds $(\mathsf{sd}_i)_{i\neq i'_0}$ and the permutation π' from the transcript T'_1 (which are the same as in T'_0). This yields

$$y^{(0)}_{d,0} = H \cdot (\pi'(U^{(0)}_{d,0}) \downarrow z^{(0)})$$
$$y^{(1)}_{d,1} = H \cdot (\pi'(U^{(1)}_{d,1}) \downarrow z^{(1)}),$$

where $z^{(0)}$ and $z^{(1)}$ are the Round 5 vectors included in the transcripts T'_0 and T'_1, and

$$z^{(0)}_{d,0} = X^{(0)}_{d,0} - \pi'(R^{(0)}_{d,0})$$
$$z^{(1)}_{d,1} = X^{(1)}_{d,1} - \pi'(R^{(1)}_{d,1}).$$

Now, because T'_0 and T'_1 share the same states $(\mathsf{state}'_i)_{i\leq n}$, it holds by construction that $U^{(0)}_{d,0} + U^{(1)}_{d,1} = u$, $X^{(0)}_{d,0} + X^{(1)}_{d,1} = x$, $R^{(0)}_{d,0} + R^{(1)}_{d,1} = r$, and $y = y^{(0)}_{d,0} + y^{(1)}_{d,1}$. Furthermore, by the collision-resistance of H_2, it must hold that $y^{(0)}_{d,b} = y^{(1)}_{d,b}$ and $z^{(0)}_{d,b} = z^{(1)}_{d,b}$ for every $b \in \{0, 1\}$. The latter equality implies that $z^{(0)} = z^{(0)}_{d,0} + z^{(0)}_{d,1} = z^{(1)}$ (this value is denoted z from now on). This gives

$$z = z^{(0)}_{d,0} + z^{(1)}_{d,1} = X^{(0)}_{d,0} - \pi'(R^{(0)}_{d,0}) + X^{(1)}_{d,1} - \pi'(R^{(1)}_{d,1}) = x - \pi(r).$$

Furthermore,

$$y = y^{(0)}_{d,0} + y^{(1)}_{d,1} = H \cdot (\pi'(U^{(0)}_{d,0} + \pi'(U^{(1)}_{d,1}) \downarrow z) = H \cdot (\pi'(u) \downarrow z).$$

By observing that $\pi(u) \downarrow z = \pi'(u) \downarrow (x - \pi'(r)) = \pi'(u \uparrow v) \downarrow x = \pi'(v) \downarrow x$, it follows that $H \cdot (\pi'(v) \downarrow x) = y$, which is a contradiction since the sampling on π' is conditioned on $\neg\mathsf{BadPerm}$. Hence, assuming the collision-resistance of H_2, it necessarily holds that $\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}} | R = R^* \wedge \neg\mathsf{BadPerm}] \leq 1/n$. Finishing the proof:

$$\begin{aligned}\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}} | R = R^*] &\leq \mathsf{p} + (1 - \mathsf{p}) \cdot \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}} | R = R^* \wedge \neg\mathsf{BadPerm}] \\ &\leq \mathsf{p} + (1 - \mathsf{p}) \cdot \frac{1}{n} = \varepsilon,\end{aligned}$$

contradicting the initial assumption that R^* is good. Therefore, a vector v, a tuple x, and a permutation π' such that $H \cdot (\pi'(v) \downarrow x) = y$ have been extracted, yet $v \in \mathbb{F}_2^K \setminus \mathsf{PN}_B$.

The extractor. Equipped with the above analysis, an extractor $\mathcal{E}$ is described, which is given rewindable black-box access to a prover $\tilde{\mathsf{P}}$. Define $N \leftarrow \ln(2)/((1-\alpha)\tilde{\varepsilon}-\varepsilon)$. $\mathcal{E}$ works as follows:

- Run $\tilde{\mathsf{P}}$ and simulate an honest verifier V to get a transcript T_0. Restart until T_0 is a successful transcript.
- Repeat N times:
  - Run $\tilde{\mathsf{P}}$ with an honest V and the same randomness as in T_0 to get a transcript T_1.
  - If T_1 is a successful transcript with $i_0 \neq i_1$, extract the tuple (x,u,r) and the permutation π. Output $\pi(v)\downarrow x$.

The end of the proof is perfectly identical to the analysis in [FJR22, Appendix F]: given that $\mathcal{E}$ found a first successful transcript T_0

$$\begin{aligned}\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}^{T_1} \wedge i_1 \neq i_0 | R \text{ good}] &= \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}^{T_1} | R \text{ good}] - \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}^{T_1} \wedge i_1 = i_0 | R \text{ good}] \\ &\geq (1-\alpha)\tilde{\varepsilon} - 1/n \geq (1-\alpha)\tilde{\varepsilon} - \varepsilon,\end{aligned}$$

Hence, by definition of N, $\mathcal{E}$ gets a second successful transcript with probability at least $1/2$. From there, the analysis of the expected number of calls $\mathbb{E}[\mathsf{call}]$ of $\mathcal{E}$ to $\tilde{\mathsf{P}}$ is identical to [FJR22, Appendix F]:

$$\begin{aligned}&\mathbb{E}[\mathsf{call}] \leq 1 + (1 - \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}]) \cdot \mathbb{E}[\mathsf{call}] + \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}] \cdot (N + (1-\alpha/2) \cdot \mathbb{E}[\mathsf{call}]) \\ \implies\ &\mathbb{E}[\mathsf{call}] \leq \frac{2}{\alpha\tilde{\varepsilon}} \cdot \left(1 + \tilde{\varepsilon} \cdot \frac{\ln(2)}{(1-\alpha)\tilde{\varepsilon} - \varepsilon}\right),\end{aligned}$$

which gives an expected number of calls $\mathsf{poly}(\lambda)(\lambda, (\tilde{\varepsilon}-\varepsilon)^{-1})$ by setting $\alpha \leftarrow (1-\varepsilon/\tilde{\varepsilon})/2$ (corresponding to $(1-\alpha)\tilde{\varepsilon} = (\varepsilon+\tilde{\varepsilon})/2$). This concludes the proof.

From soundness to EUFKO security. Given a five-round identification protocol where the probability of sampling a "bad" Round 3 challenge is bounded by p, and the probability of sampling a "bad" Round 5 challenge is bounded by $1/n$, it follows from a standard application of the Fiat-Shamir methodology (adapted to 5-round protocols) to the τ-fold parallel repetition of the identification scheme given in Protocol 17 that, when modeling H_1 and H_2 with random oracles, there exists an extractor which extracts a weakly valid witness $x^* \in X$ given any adversary that succeeds with probability at least $\Pr[X+Y=\tau]$, where with probability at least $1-\varepsilon_{\mathsf{G}}$ over the random choice of (H,y), it holds that $X = \max_{\alpha\in Q_1}\{X_\alpha\}$ and $Y = \max_{\beta\in Q_2}\{Y_\beta\}$ with $\mathsf{p} = 4/B$, $X_\alpha \sim \mathsf{Binomial}(\tau,\mathsf{p})$, and $Y_\beta \sim \mathsf{Binomial}\left(\tau - X, \frac{1}{N}\right)$ where Q_1 and Q_2 are sets of all queries to the oracles H_1 and H_2. With probability at least $1-1/2^\lambda$, this weakly valid witness is necessarily a regular witness. This concludes the proof. □

Chapter 7

Threshold Signatures from MPC-in-the-Head

As discussed in Section 4.2, threshold signatures offer resilience to failures, data loss, and enhance decentralization. Despite the success of MPC-in-the-Head in enabling efficient schemes [KKW18; KZ20b; BDK+21; ZCD+20; DDO+19; FJR22; AGH+23; BKP+23; ABC+23; ABB+23; BFR23; GSR23; DGO+23; CHT23; HJ24; BBS+23; LMØ+23; ZLH24; BBM+24; OTX24], extending it to threshold signatures remains an open problem: current threshold schemes [PKM+24] fail to achieve efficiency when built on MPC-in-the-Head due to reliance on GGM trees and the need for non-black-box use of cryptographic primitives. Recent work [DKR24] proves that threshold MPC-in-the-Head signatures must either tolerate large signature sizes or use non-black-box techniques, highlighting inherent limitations in the current approaches. In this chapter, a threshold signature is defined starting from the signature presented in Chapter 5, circumventing all the challenges arising from its structure.

Contents

From now on, the term *parties* will be used to refer to the virtual participants emulated in its head by the prover of an MPC-in-the-Head protocol, and the term *users* will refer to the actual participants that interact to distributively generate a signature. Given a vector x of length $6n$, viewed as a concatenation of n "blocks" of size 6, let $\mathsf{BHW}_6(x)$ denote the vector over $\mathbb{F}_3$ whose i-th entry is the Hamming weight of the i-th block of x modulo 3.

7.1 Problem Statement

In this chapter, the feasibility of designing efficient threshold signatures from MPCitH signatures is explored, despite the strong impossibility result of [DKR24]. The main contribution is a general recipe for converting any MPCitH signature into a "threshold-friendly" signature scheme. This approach does not circumvent the impossibility result of [DKR24]: instead, it accepts that *the size of the signature scheme grows with the number of users*, by considering the following question: given that [DKR24] implies that all threshold-friendly MPCitH signatures must grow with the number of users (or else make expensive non-black-box use of cryptography), how much can this linear dependency in the number of users be reduced? The main finding is the following: there is a general template for transforming any MPCitH signature into a threshold-friendly signature of size $\lambda^2 n + O(1)$ bits, where the constant depends on the concrete scheme. When $\lambda = 128$, this results in a size of 2kB per user. This represents a significant improvement over the naive strategy of concatenating independent signatures from all users. While these numbers may seem somewhat underwhelming and are not competitive with lattice-based threshold signatures such as [PKM+24], they provide a useful data point regarding the best-possible signature size achievable in a threshold-friendly setting. It is hoped that this result will inspire further work in the area.

An interesting consequence of this result is that, in the context of threshold signatures, the size difference between existing MPCitH candidates is reduced to the $O(1)$ term, which becomes negligible as n grows. This has the conceptually intriguing effect of inverting the notion of the "best" MPCitH signature in this context: older schemes, which are no longer state-of-the-art and have larger sizes, tend to have simpler structures. This simplicity results in much more efficient threshold signature schemes, particularly in terms of the communication and computation required for distributively generating a signature. This behavior is illustrated through a detailed case study applying the template to the MPCitH scheme from Section 5.4, a scheme based on the regular syndrome decoding assumption. Although it has been superseded by more recent works [ZLH24; OTX24] that achieve significantly more compact signatures, its simple structure allows for a very straightforward and efficient distributed signing procedure in its threshold-friendly variant.

To provide concrete numbers: using $K = 1842$, $k = 1017$, $w = 307$, $\tau = 11$, and $N = 2^{13}$ from Section 5.4, the threshold signature scheme achieves amortized communication of 71.2kB per user (when instantiating the underlying "modulus-switching" functionality as described in Section 7.5.3.1) and produces signatures of size approximately $2n + 6$ kilobytes.

7.2 Challenges in thresholdizing MPCitH signatures

7.2.1 An MPC-in-the-Head and VOLE template.

The starting point is the observation that all modern MPC-in-the-Head and VOLE-in-the-Head signatures share a common high-level template (up to, sometimes, minor variations). A rough outline of the template is as follows:

- The signer, using a root key and some random salt, derives root keys and salts for each round (where the rounds correspond to the number of repetitions τ of the underlying identification scheme to achieve negligible soundness error).
- For each round $k \leq \tau$, using the round root key and salt, the signer expands the root key into $N = 2^d$ pseudorandom (leaf seed, commitment) pairs, denoted $(\mathsf{sd}_i^k, \mathsf{com}_i^k)$, using a full binary tree *a la* GGM, where each internal node has two children, computed by applying a length-doubling pseudorandom generator on the value of the parent node.
- The concatenation of all commitments of the rounds is hashed with a collision-resistant hash function, and all hashes of all rounds are hashed together into a single hash.
- For each round, the leaf seeds are further expanded via a PRG into 2^d pseudorandom additive shares $(s_j^k)_{j \leq N}$ of a target tuple, which usually consists of the witness together with (potentially) some correlated random coins. An auxiliary string, or *shift*, is constructed to offset the sum of the shares such that it reconstructs to the target tuple (alternatively, the shifts can be computed after the collapsing step that follows).
- For each round k, the 2^d shares are collapsed into d aggregated values, where the i-th value is computed by aggregating all shares s_j^k such that the i-th bit of j (viewed as a bitstring over $\{0,1\}^d$) is 0. The MPC-in-the-Head view on this process is that of collapsing one 2^d-party (virtual) MPC instance into d 2-party (virtual) MPC instances by aggregating the shares along the hyperplanes of the d-dimensional boolean hypercube. The VOLE-in-the-Head view differs in semantics (the collapsing process is viewed as a reduction from an $(N-1)$-out-of-N oblivious transfer instance to a (subspace) vector-OLE correlation over the subspace $\mathbb{Z}_d$), but the process is identical.
- (Optional) A *consistency challenge* (for each round) is derived from the shifts and the hash of the commitments. This challenge is used to check the consistency of the shifted aggregated shares concerning the correlation they are supposed to satisfy.
- (Optional) A proof of consistency is derived from the challenge and the aggregated shares. Usually, this proof has a very simple structure and involves only challenge-dependent linear combinations of the aggregated shares.

- A *protocol challenge* (for each round) is derived from the shifts and the hash of the commitments (or, if a consistency challenge was used, from the consistency challenge and the consistency proof). This challenge is used for the execution of the main virtual MPC protocol (for MPC-in-the-Head proofs) or VOLE-based ZK proof (for VOLE-in-the-Head proofs) and is typically a "Schwartz-Zippel" challenge (used to collapse the verification of a set of multivariate polynomial equations).
- For each round, the virtual MPC protocol, or the designated-verifier VOLE-based ZK proof, is run.
- An *opening challenge* for each round is derived from the previous challenge. This challenge is used to define the virtual parties to open in the MPC-in-the-Head protocol (or, in VOLE-in-the-Head language, it defines the VOLE MAC key for the designated-verifier VOLE-based ZK proof).
- Eventually, an opening of each round is computed. It contains the opening to all-but-one of the leaf seeds (if the challenge is $\Delta_k \in \{0,1\}^d$ in round k, the opening contains the values on the nodes of the co-path from the root to the leaf Δ_k in the k-th GGM tree) as well as $\mathsf{com}^k_{\Delta_k}$.

On Figure 7.1, a more precise description of the typical MPC-in-the-Head template is represented, indicating the subroutines involved, their inputs and outputs, and using colors to provide a high-level overview of the difficulties that arise in a threshold setting. The term "PPRF" stands for puncturable pseudorandom function [KPT+13; BW13; BGI14] and captures the abstract primitive realized by the GGM PRF [GGM86]: a pseudorandom function with domain size N equipped with an efficient puncturing algorithm that, given a PPRF key K and a point $\Delta \leq N$, generates a succinct key $K\{\Delta\}$ that allows recomputation of the PPRF on all points except Δ. "VOLE" stands for vector oblivious linear evaluation, and denotes the (linear) aggregation procedure that converts a 2-party $(N-1)$-out-of-N oblivious transfer into additive shares of $\Delta \cdot u$, where Δ is the punctured point (the selected leaf) and u is a pseudorandom string known to the signer. This step is syntactically identical to the hypercube technique [AGH+23] that collapses an N-party virtual MPC protocol into $d = \log N$ instances of a 2-party protocol (but the alternative VOLE-style view has proven conceptually useful and is at the heart of the line of work on VOLE-in-the-Head signatures). The color code of Figure 7.1 is outlined below:

▭ : the red color denotes the subroutines that are challenging to generate in a distributed setting, and whose naive distributed evaluation is highly non-black-box. These are handled by duplicating the PPRF instances, letting each user locally run its own PPRF instances and computing the VOLEs / hypercube aggregated shares. The VOLEs obtained by each user are viewed as additive shares of the global VOLE of the signature.

▭ : the blue color denotes calls to a hash function on the private outputs of the PPRF functionality. Instead, each party locally hashes its shares of the commitments, and all

parties hash the concatenation of the local hashes.

: the green color denotes functionalities that are typically much simpler to evaluate distributively (often involving mostly linear operations). When adapting the high-level template to a concrete choice of MPCitH signature, the bulk of the work consists of designing efficient MPC protocols for computing the green subroutines.

: the gray color denotes the optional Quicksilver proof challenge [YSW+21]. This check is typically present in VOLE-in-the-Head protocols [BBS+23; LMØ+23; ZLH24; BBM+24; OTX24], but absent from MPC-in-the-Head protocols [KZ20b; BDK+21; ZCD+20; DDO+19; FJR22; AGH+23; BKP+23; ABC+23; ABB+23; BFR23; GSR23; DGO+23; CHT23; HJ24] that directly rely on a virtual MPC protocol to prove the target relation (though, in principle, such a protocol can also receive a challenge, typically to introduce batching and make verification more efficient).

→ : the red arrow denotes outputs that are directly appended to the signature or that can be computed from values included in the signature.

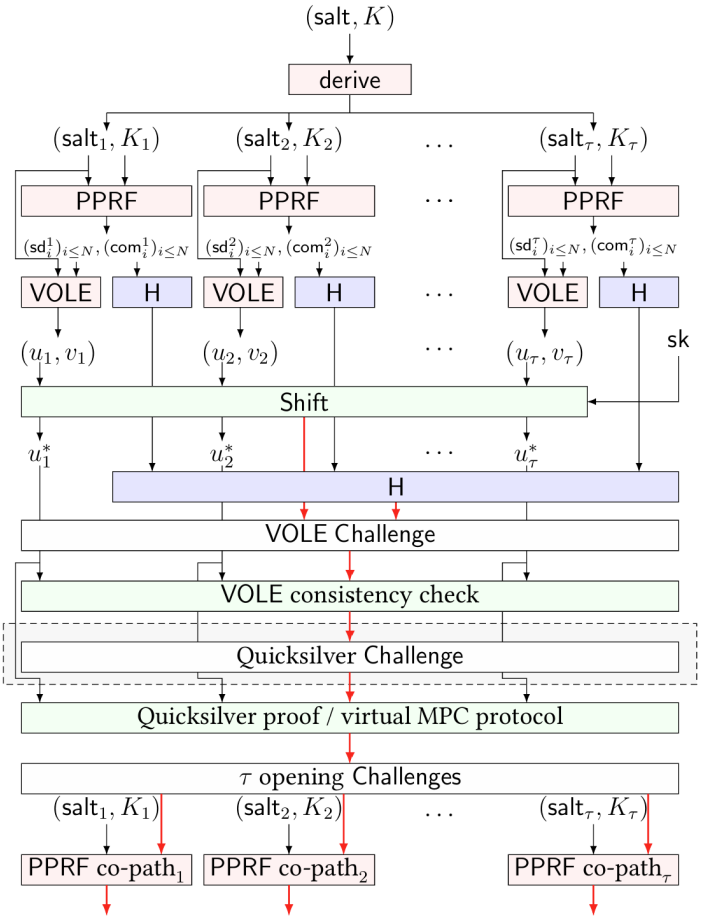

Figure 7.1: small High-level representation of the common structure of existing MPC-in-the-Head and VOLE-in-the-Head signature schemes. sk *denotes the secret key, or* witness *(the preimage of some one-way function).*

7.2.2 Inefficiencies in thresholdizing MPCitH signatures

Any signature scheme can be generically converted into a threshold signature scheme via generic maliciously secure MPC. However, the structure of MPC-in-the-Head signature schemes makes them especially ill-suited for such generic approaches, resulting in extremely inefficient constructions.
The main challenge lies in the construction of the GGM tree, since to generate a signature:

- The signer samples τ root keys (and salts) and generates τ full binary trees using a length-doubling PRG (e.g., SHA3 [CHT23], AES in counter mode [LMØ+23], or fixed-key AES 6.3).
- Each tree has N leaves, where N typically ranges from 2^8 ("fast") to 2^{16} ("short"). Figure 7.2 shows an example with $N = 2^4$.
- Each leaf is stretched (via a PRG) into a *commitment* and a *virtual party share*, represented respectively in green and purple on Figure 7.2.
- An auxiliary string is constructed from the aggregated N virtual party shares to correct the N-th share, and the N commitments are concatenated and hashed to create a succinct commitment.
- Once the challenge leaf i is determined, a succinct opening to all leaves except i is added to the signature by including all seeds on the co-path to the selected leaf node (the co-path is represented in blue on Figure 7.2, and the selected leaf in red).

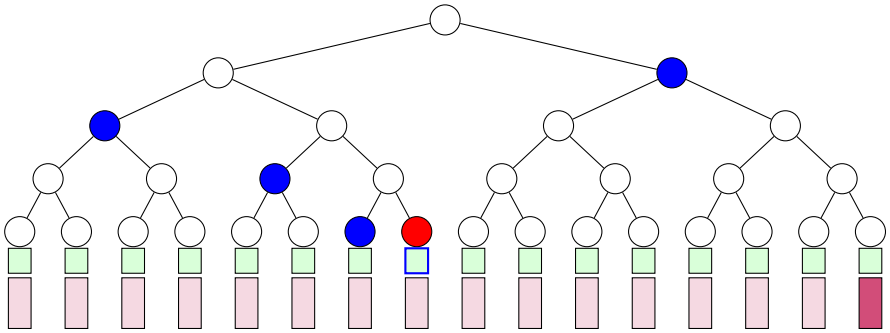

Figure 7.2: A full GGM tree with $N = 16$. The seeds on the leaves are stretched into two strings, a commitment string (in green) and a virtual party share (in purple) The red-colored leaf denotes the seed that is not revealed in the signature, and the blue nodes denote the seeds on the co-path to the selected leaf. The signature contains the co-path seeds as well as the commitment string to the red leaf (denoted by a green-filled blue rectangle). The N-th virtual party share, represented in a darker purple, is computed from the aggregated virtual party shares from 1 to $N - 1$ instead of being pseudorandomly generated from the N-th leaf seed.

7.2.2.1 Limitations of the naive approach To distributively generate an MPC-in-the-Head signature, users must perform the following steps:

1. Run (τ times in parallel) a maliciously secure protocol that expands shares of a root seed into a full binary tree of PRG evaluations, and stretches each leaf seed via a PRG, ensuring the virtual party shares remain secret.
2. Reveal the N commitments (which can be locally hashed into a short digest).
3. Securely compute the N-th virtual party share (auxiliary string) from the aggregated virtual party shares of the previous $N - 1$ leaves.
4. Once the non-opened leaf node is determined, reconstruct and append to the signature the seeds on the co-path to the selected leaf.

The above procedure is extremely inefficient: even in the two-party setting, a state-of-the-art maliciously secure constant-round protocol for AES [WRK17a] requires, on average, 6.7ms per AES computation and 5.2MB of communication. Each of the τ full binary trees contains $2N$ nodes, with at least two additional AES evaluations per leaf node (to generate the commitment and virtual party share). This results in a total of $4\tau N$ AES evaluations. For FAEST-128s parameters [LMØ+23], where $N = 2^{12}$ and $\tau = 11$, the total communication reaches nearly a terabyte (937GB), with more than 20 minutes of computation required only to expand the binary trees. As the number of users increases, the inefficiencies grow significantly worse. At a higher level, two main issues arise:

(1) The protocol relies on a *non-black-box use of the PRG*, resulting in significant computational costs.

(2) Communication scales with N, defeating the purpose of MPCitH, where large N usually reduces communication.

7.2.2.2 The black-box barrier Could the non-black-box use of PRGs be avoided? Unfortunately, the answer is no: Doerner, Kondi, and Rosenbloom [DKR24] proved that any threshold MPC-in-the-Head signature scheme must either make a non-black-box use of hash functions/PRGs, or tolerate a signature size that grows with the number of signers. This impossibility result directly applies to schemes like FAEST and Picnic, showing that the naive approach cannot be significantly improved without modifying the signature scheme.

7.2.3 The threshold-friendly approach

The impossibility result of Doerner et al. [DKR24] is quite strong, and no way to circumvent it is evident. As a result, the MPC-in-the-Head approach is modified to make it *threshold-friendly*, allowing the size of the signature to grow with a bound on the number of signers.

7.2.3.1 Scaling the GGM trees A trivial way to make a signature scheme threshold-friendly is to increase its size proportionally with the number of users: given n signing keys $(\mathsf{sk}_1, \cdots, \mathsf{sk}_n)$, a threshold signature could be defined as:

$$\mathsf{Sign}'(m, (\mathsf{sk}_1, \cdots, \mathsf{sk}_n)) := \mathsf{Sign}(m, \mathsf{sk}_1) || \cdots || \mathsf{Sign}(m, \mathsf{sk}_n).$$

In this case, the computation among n users is straightforward: each user samples their own key and locally generates their signature. However, in the context of MPC-in-the-Head signatures, a more efficient approach can be achieved by letting only the number of *GGM trees* scale with the number of users.
The MPC-in-the-Head signature consists of τ co-paths, where each co-path corresponds to the seeds on the path from the root node of a GGM tree to the selected leaf. The total size of these co-paths is independent of N and is identical for nearly all existing MPC-in-the-Head schemes. Specifically:

$$\text{Co-path size} = \lambda \cdot \log N, \quad \tau = \frac{\lambda}{\log N}.$$

Here, $\lambda = 128$ ensures soundness $1/2^{\lambda}$ after τ repetitions, and the total size of the co-paths becomes $\tau\lambda \log N \approx \lambda^2$ (*i.e.* 2kB for $\lambda = 128$).
This leads to a significant reduction in size compared to naively concatenating n signatures. For example:

- FAEST-128s and FAEST-128f signatures (5kB and 6.3kB respectively) result in signature sizes of $2n + 3$ kB and $2n + 4.3$ kB for n users. This is much smaller than the naive $5n$ kB and $6.3n$ kB sizes.
- The scheme from 5.4 (12.5kB and 9kB for fast and short variants) achieves sizes of $2n + 10.5$ kB and $2n + 7$ kB respectively, compared to the naive $12.5n$ kB and $9n$ kB.

As a consequence, the difference in signature sizes among various MPC-in-the-Head schemes becomes asymptotically negligible for large n. This shifts the focus to the computational and communication costs of generating signatures distributively. For example, some schemes such as the one in Section 5.4, which are no longer state-of-the-art in terms of signature size, perform well in this setting, while state-of-the-art schemes like FAEST [LMØ+23] face greater challenges in achieving reasonable efficiency.

7.2.3.2 The proposed approach In more detail, the proposed approach generates n independent GGM trees from n independent roots for each of the τ rounds. The pseudorandom strings computed from the seed leaves of each tree are viewed as *n-user shares of the commitments and virtual party shares*. Specifically *the i-th virtual party commitment and share are defined as the XOR of the i-th leaf strings of each of the n GGM trees.*
This design enables each user to locally generate their own GGM tree, ensuring that the virtual party shares are held as n-wise shares without requiring any communication.

Distributively computing a signature then reduces to the following four tasks:

1. Compute the hash of the concatenation of all commitments.
2. Distributively compute the auxiliary string.
3. Distributively compute the virtual MPC protocol.
4. Append all $\tau \cdot n$ co-paths to the final signature.

Items 2 and 3 can typically be performed efficiently. The auxiliary string is computed via a distributed protocol from the locally-aggregated strings of each of the n users, ensuring that the total communication remains independent of the number N of leaves. Additionally, the virtual MPC protocol operates on $d = \log N$ virtual party shares (collapsed from N shares via the hypercube technique [AGH+23]), and its extension to $n \cdot N$ parties introduces minimal overhead in many cases.
Item 4, on the other hand, increases the signature size by $2(n-1)$ kB when $\lambda = 128$. This linear-in-n growth is unavoidable for schemes whose threshold version makes a black-box use of the underlying primitive, as discussed earlier.[1]
For item 1, reconstructing the N commitments for each of the τ rounds (Item 1) imposes a significant communication cost. For example:

- 5.8MB per user for $\tau = 11$ and $N = 2^{12}$,
- 75.5MB per user for $\tau = 9$ and $N = 2^{16}$.

To mitigate this overhead, the following optimization is applied:

- Each user locally hashes the concatenation of their commitment shares and broadcasts the resulting hash.
- The final hash is obtained by concatenating all n hashes.

This optimization reduces communication to *32 bytes per user*, with a slight increase in signature size by $16n$ bytes, as all n shares of the selected leaf commitment must be included for verification. This transform reduces the size gap between MPC-in-the-Head schemes, shifting the focus to computational and communication efficiency in the threshold setting.

7.3 Threshold signatures from threshold-friendly MPCitH signatures

Given a threshold-friendly MPCitH signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ that follows the template outlined in the previous section 7.2.1, a distributed signing procedure is sketched.

[1] Lowering this cost below 2kB/user remains an open challenge.

Let Commit denote an extractable and equivocable commitment scheme. The notations from the template of Figure 7.1 are used. The protocol involves n users $\mathcal{U}_1, \cdots, \mathcal{U}_n$. It is assumed that the setup is executed by a trusted dealer.[2]

Trusted setup.

1: Sample $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
2: Sample n shares $(\mathsf{sk}_1, \cdots, \mathsf{sk}_n)$ of sk.
3: Output pk as a public output, and send sk_j to each user $\mathcal{U}_j$. // Additionally, the trusted setup might append other correlated material to facilitate the executions of distributed protocols during the signing phases, such as e.g. PCG seeds.

Commit-and-open phase.

1: Each user $\mathcal{U}_j$ samples τ root keys $(K_{1,j}, \cdots, K_{\tau,j})$.
2: A master salt salt is sampled via a secure coin-flipping protocol, and τ salts $(\mathsf{salt}_1, \cdots, \mathsf{salt}_\tau)$ are derived from salt using a PRG modelled as a random oracle.
3: Each user $\mathcal{U}_j$ locally computes $(\mathsf{sd}_i^{e,j}, \mathsf{com}_i^{e,j})_{i \le N} \leftarrow \mathsf{PPRF}(\mathsf{salt}_e, K_{e,j})$ for $e = 1$ to τ and sets $h^j \leftarrow \mathsf{H}(\mathsf{com}_1^{1,j}, \cdots, \mathsf{com}_N^{1,j}, \cdots, \mathsf{com}_1^{\tau,j}, \cdots, \mathsf{com}_N^{\tau,j})$.

Commit: each user $\mathcal{U}_j$ sends $c_j \leftarrow\!\!\$\ \mathsf{Commit}(h^j)$.

Open: each user $\mathcal{U}_j$ opens c_j to h^j.

Hash: all users compute $h_1 \leftarrow \mathsf{H}(h^1, \cdots, h^n)$.

Shift.

1: Each user $\mathcal{U}_j$ generates τ VOLE pairs $(u_{e,j}, v_{e,j}) \leftarrow \mathsf{VOLE}(\mathsf{sd}_1^{e,j}, \cdots, \mathsf{sd}_N^{e,j})$ (equivalently, τ pairs of hypercube-aggregated shares).
2: For every e, the pairs $u_{e,j}, v_{e,j})_{j \le n}$ are viewed as additive shares of a single VOLE pair (u_e, v_e).
3: All parties engage in a maliciously secure MPC protocol to securely instantiate the n-party functionality $\mathcal{F}_{\mathsf{shift}}$ that takes as input shares of $(u_e, v_e)_{e \le \tau}$ and publicly outputs the shift shift (in VOLE-in-the-Head terminology) or auxiliary string aux (in MPC-in-the-Head terminology).

VOLE consistency check.

1: All parties derive from (h_1, shift) a first challenge chall_1.
2: In the VOLE-in-the-Head setting ser $\mathcal{U}_j$ locally computes shares of the VOLE consistency check from $(\mathsf{chall}_1, \mathsf{shift}, (u_{e,j}, v_{e,j})_{e \le \tau})$. The users broadcast their shares and reconstruct the VOLE consistency check (this is possible because the check is performed via a linear universal hash function).
3: In the MPC-in-the-Head setting, the auxiliary string typically has a more complex structure (e.g. Beaver triples in [BDK+21; CHT23], shares of the same pseudorandom bit over $\mathbb{F}_2$ and $\mathbb{F}_3$ in Chapter 5, or shares of a pseudorandom integer and its one-hot-vector rep-

[2] This is a reasonable assumption in the common scenario where a signer delegates its signing capability to n untrusted servers. In other contexts, no signer is assumed to know the full secret key, and the trusted setup must be replaced by a distributed protocol. For simplicity, the former scenario is considered in this work.

resentation in Chapter 6). Accordingly, the check becomes more involved. Nevertheless, in existing MPC-in-the-Head protocol, this consistency check remains fully linear (it involves sacrificing in [BDK+21; CHT23] and randomly shuffling the correlated randomness in 5.4 and 6.3 using a public random permutation derived from the challenge).

(Optional) Quicksilver challenge.

1: All parties publicly derive from (h_1, shift) and from the VOLE consistency check a challenge chall_2 for the Quicksilver proof.

Quicksilver proof / virtual MPC protocol.

1: In this step, the parties distributively run the virtual MPC protocol (alternatively, distributively generate the quicksilver proof). The cost of this step varies widely from one signature scheme to the other: in some signature schemes such as the ones presented in this manuscript, the virtual MPC protocol is reduced to a bare minimum, involving only local linear operations and broadcasting shares. As a consequence, distributively emulating this virtual MPC protocol among n users is straightforward. In contrast, in schemes such as FAEST (and all VOLE-in-the-Head schemes), this step requires executing a maliciously secure protocol for distributively generating a Quicksilver proof, which appears considerably more challenging. In particular, in FAEST, it requires running a maliciously secure n-user evaluation of the AES circuit over an extension field.[a]

Opening.

1: Eventually, all parties publicly derive from (h_1, shift) and the transcript π of the virtual MPC protocol / the Quicksilver proof π a challenge chall_3 that encodes the τ positions $(i_1, \cdots, i_\tau) \in [N]^\tau$ that should not be opened.

2: Each user $\mathcal{U}_j$ computes $(\mathsf{CoPath}^{e,j}, \mathsf{com}^e_{i_e})_{e\le\tau}$, where $\mathsf{CoPath}^{e,j}$ denotes the $\log N$-sized tuple of seeds on the co-path from the root $K_{e,j}$ to the i_e-th leaf.

3: They output the final signature

$$\sigma = (\mathsf{salt}, h_1, (\mathsf{chall}_2), \mathsf{chall}_3, (\mathsf{CoPath}^{e,j}, \mathsf{com}^e_{i_e})_{e\le\tau, j\le n}, \pi, \mathsf{shift}).$$

[a] The work of [WRK17b] reports more than 20s of computation for 14 parties, and about 3 minutes for 128 parties, when evaluating the plain AES circuit; the version needed here would be a large constant factor larger, since the AES circuit must be evaluated over an extension field.

Protocol 18: A template protocol for an MPC/VOLE threshold signature

7.4 Security and Performance Analysis

Obstacles towards proving security. The above template threshold signature appears intuitively secure. However, security cannot be reduced to the unforgeability of the underlying (threshold-friendly) MPC-in-the-Head scheme. To understand the issue, consider the first steps of the security analysis. Assume that all users except $\mathcal{U}_j$ are corrupted, and let Sim denote a simulator that emulates $\mathcal{U}_j$. In the setup phase, Sim receives pk from the signature functionality and samples $n-1$ random shares $(\mathsf{sk}_\ell)_{\ell\neq j}$. During the commit-and-open phase,

Sim commits to a dummy string (its opening can be adapted later since the commitment scheme is equivocable) and extracts $(h^\ell)_{\ell \neq j}$ from the commitments $(c_\ell)_{\ell \neq j}$.
At this point, the proof becomes essentially stuck. The goal of Sim is to act as an interface between the corrupted parties and the signing oracle. To do so, Sim is not given access to the secret key and must simulate using only access to the signing oracle. By the unforgeability of the signature scheme, this implies that the only way for Sim to properly simulate a threshold signing session on a message m is to somehow force the signing session to output the exact same signature received from the signing oracle.[3]
Given access to a signing oracle, Sim receives a signature $\sigma = (\mathsf{salt}, h_1, (\mathsf{chall}_2), \mathsf{chall}_3, (\mathsf{CoPath}^{e,j}, \mathsf{com}^e_{i_e})_{e \leq \tau, j \leq n}, \pi, \mathsf{shift})$ on a message m. From the signature, Sim obtains $h_1 = \mathsf{H}(\tilde{h}^1, \cdots, \tilde{h}^n)$. To simulate, Sim would need to find h^j such that $\mathsf{H}(h^1, \cdots, h^n) = \mathsf{H}(\tilde{h}^1, \cdots, \tilde{h}^n)$, which would break the collision-resistance of H (and thus the security of the signature scheme). Worse, the co-paths and commitments $(\mathsf{CoPath}^{e,j}, \mathsf{com}^e_{i_e})$ included in the signature allow recomputing the $\tilde{h}^\ell$. Because Sim must force the signing session to output the same co-paths as contained in σ, the adversary would inevitably notice in the final signature that the $\tilde{h}^\ell$ recomputed from the co-paths are not equal to the h^ℓ. Therefore, Sim cannot simulate the interaction at this stage.

Candidate workarounds. This issue arises directly from the handling of item 1 in Section 7.2.3.2. If the parties computed h_1 as a hash of the XOR of their individual commitment shares, then Sim could reconstruct from σ the commitment tuple $(\mathsf{com}^e_i)_{i \leq N, e \leq \tau}$, extract the commitment shares of the corrupted users from $(c_\ell)_{\ell \neq j}$, and adapt the opening of its commitment c_j to the XOR of the commitment tuple and the corrupted parties' commitment shares. This strategy is similar to the one used in standard threshold signature schemes, such as threshold variants of Schnorr. However, in the MPCitH context, this approach would incur prohibitive communication overhead.
To preserve the low communication of the current solution, the most natural workaround would be to modify the threshold signing protocol to enable Sim to force the corrupted users to output the same hashes $\tilde{h}^\ell$ as contained in σ. The only way to achieve this would be to compute the hash of each user via a distributed protocol instead of locally. However, computing the hash values h^ℓ via a distributed protocol would again result in prohibitive communication overhead, as the input size to each hash is very large, negating the communication advantages of the method. Furthermore, such a distributed computation would require non-black-box use of H, introducing significant computational complexity.

[3] Technically, Sim could potentially force the output of a different signature if the signature scheme is unforgeable but not strongly unforgeable. However, as all known MPCitH schemes are plausibly strongly unforgeable, it seems unlikely that this strategy could be implemented.

7.4.1 Corrupted Existential Unforgeability under Chosen Message Attack

To resolve this conundrum, a different strategy is adopted: the threshold signing protocol is not changed; instead, the *unforgeability game* of the signature is modified. The main observation, which is very natural in retrospect, is the following: in the course of modifying the signature scheme to make it threshold-friendly, the scheme also becomes *more secure* in a specific sense that will be clarified shortly. The security proof of existential unforgeability of an MPCitH signature relies at some point on a sequence of game hops to replace the co-path and the selected leaf seed with uniformly random strings, by invoking $\log n$ times the security of the PRG. In turn, this allows the replacement of the share generated from the selected leaf with a random string. Since a portion of this string is used to mask the witness, this guarantees that the witness remains hidden (statistically, at this point).
Now, in the threshold-friendly variant, there are n GGM trees and n co-paths, and the $n \cdot N$ strings stretched from the leaf seeds of all trees form additive shares of the N virtual parties' shares. The same analysis as with a single co-path applies to this setting, but crucially, it suffices to replace the selected leaf share of a *single GGM tree* with a random string (this is easy to see: the "shares of shares" form a $n \cdot N$-party additive sharing of the witness, and replacing a single share with a random string suffices to statistically mask the witness). This implies that the scheme satisfies the following stronger security property: it remains secure even if *all but one* of the root keys are controlled by the adversary (instead of being randomly sampled by the signing oracle), as long as one root key is guaranteed to remain honestly and secretly sampled by the signing oracle. In other words, the threshold-friendly variant achieves a form of resistance against partial corruption of the random tape of the signature scheme.
This observation is formalized in Definition 7.4.1 by introducing the notion of *corruptible existential unforgeability*. Informally, a signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ satisfies (n-users) corruptible existential unforgeability against chosen-message attacks (CEUF-CMA) if the randomness of Sign is an n-tuple $(r_1, \cdots, r_n)$, and no adversary can forge a signature after making an arbitrary polynomial number of queries to the following *corruptible* signing oracle: the oracle receives queries $(m, i, (r_j)_{j \neq i})$, samples r_i, and returns $\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}; (r_1, \cdots, r_n))$.
Returning to the security analysis of the threshold signature scheme, Sim can emulate the commit-and-open phase of the protocol as follows:

- It extracts the hashes $(h^\ell)_{\ell \neq j}$ from the corrupted users' commitments.
- From each h^ℓ, Sim extracts the root keys $(K_{1,\ell}, \cdots, K_{\tau,\ell})$. This extraction step will be discussed later (but in short, it relies on modeling H as a random oracle, and the AES cipher used to instantiate the GGM PPRF as an ideal cipher, and allowing Sim to observe the queries).

- Sim sends $(m, j, (K_{1,\ell}, \cdots, K_{\tau,\ell})_{\ell \neq j})$ to the corruptible signing oracle, and receives a signature σ. By design, the hash h_1 contained in σ is of the form $\mathsf{H}(h^1, \cdots, h^j, \cdots, h^n)$, where the $(h^\ell)_{\ell \neq j}$ are the same as those extracted from the commitments c_ℓ (as they have been computed from the same root keys).
- Sim recomputes h^j from σ using $(\mathsf{CoPath}^{e,j}, \mathsf{com}^e_{i_e})_{e \leq \tau}$ and adapts the opening of c_j to h^j using equivocability.

7.4.1.1 A dummy attack Alas, the strategy described above does not yet quite work due to an annoying "dummy attack" that a corrupted party could execute. In the analysis, it was assumed that given the hashes h^ℓ extracted from $\mathcal{U}_\ell$'s commitments (where $\mathcal{U}_\ell$ denotes the ℓ-th user), Sim could recover the root keys $(K_{1,\ell}, \cdots, K_{\tau,\ell})$ of $\mathcal{U}_\ell$ by observing its queries to the random oracle and to the ideal cipher. However, what happens if Sim does *not* find preimages $(K_{1,\ell}, \cdots, K_{\tau,\ell})$ among the queries that are consistent with h^ℓ? Several reasons could explain this situation. Perhaps $\mathcal{U}_\ell$ created a query uncorrupted or simply did not query anything and sampled h^ℓ at random. These cases are easy to rule out: with overwhelming probability, the collision-resistance and one-wayness of the random oracle guarantees that the user will fail to produce an accepting signature, and Sim can proceed through the simulation using dummy values.
However, there is one specific way for $\mathcal{U}_\ell$ to avoid querying one of the root keys $K_{e,\ell}$ and still produce an accepting signature with non-negligible probability: $\mathcal{U}_\ell$ can *guess a selected leaf* i^*, sample uniformly random seeds on the co-path to i^* (the blue nodes on Figure 7.2), and sample a uniformly random seed on i^* (the red node on Figure 7.2). Using these "fake" seeds, $\mathcal{U}_\ell$ can recompute all seed leaves of the GGM tree and run the entire threshold signing protocol. At the end of the protocol, during the opening phase, a challenge i^e will be generated. With probability $1/N$, it might happen that $i^e = i^*$, in which case, by adding its fake co-path to the signature, $\mathcal{U}_\ell$ creates a valid signature! (Not an honestly distributed signature, but one that still passes verification). This is a dummy attack that achieves nothing — $\mathcal{U}_\ell$ still requires a share sk_ℓ of sk to compute the remainder of the signature unless it produces fake co-paths for *all* $e \in [\tau]$, which happens only with negligible probability. Nonetheless, this attack breaks the simulation, as Sim cannot simply assume that failing to extract a root key guarantees the signature will not be verified.
This last challenge is handled by revisiting the (corruptible) threshold-friendly signature scheme construction. The syntax of the scheme is modified to allow it to take as random input either root seeds or pairs (co-path, selected leaf seed). The signature algorithm then proceeds as before, reconstructing the leaf seeds either from the root or from the co-path. If a root $K_{e,\ell}$ is replaced with a co-path to a leaf i^* but $i^* \neq i^e$, the signing algorithm raises a flag $\mathsf{flag} = \perp$ (indicating that the signature generation failed to produce a valid signature), but still outputs the invalid signature. Fortunately, this syntactic change has almost no impact on the CEUFCMA security analysis of the scheme (as the analysis relies essentially on replacing the *uncorrupted* root seed with a random co-path to the correct selected leaf),

though it does make the description of the scheme more tedious.
With this modification, the dummy attack can be circumvented as follows: during the commit-and-open phase, Sim will attempt to extract *either* root keys $K_{e,\ell}$ or pairs (co-path, selected leaf seed) from the preimages. If neither extraction attempt succeeds, Sim proceeds with the rest of the simulation using dummy values, and the signature is guaranteed to be invalid (with overwhelming probability over the choice of the random oracle and ideal cipher). If extraction succeeds, Sim feeds the extracted values to the corruptible signing functionality (which accepts either root keys or (co-path, leaf) pairs as randomness input) and proceeds with the rest of the simulation using the signature returned by the functionality (which may or may not be a valid signature, depending on whether a corrupted party attempted a dummy guess attack and guessed incorrectly).

7.4.1.2 Corruptible existential unforgeability At this point, it is now possible to formally introduce the *Corruptible Existential Unforgeability under Chosen Message Attack* (**CEUF-CMA**) security model, in which an adversary interacts with the challenger by querying signatures on tuples $(m, i, (r_j)_{j\neq i})$, where the chosen seeds are used by the challenger for all-but-one of the trees in the MPC.

Definition 7.4.1. *Let $n = n(\lambda)$ be a polynomial, and let $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme. Σ is defined as an $(n+1)$-source signature scheme if there exist $n+1$ sets $(\mathcal{R}, \mathcal{R}_1, \cdots, \mathcal{R}_n)$ such that the random tape of Sign is sampled from the randomness domain $\mathcal{R} \times \mathcal{R}_1 \times \cdots \times \mathcal{R}_n$.*

Definition 7.4.2. *Let $n = n(\lambda)$ be a polynomial, and let $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be an $(n+1)$-source signature scheme with randomness domain $\mathcal{R} \times \mathcal{R}_1 \times \cdots \times \mathcal{R}_n$. Consider the following experiment $\mathsf{Exp}^{\mathsf{CEUFCMA}}_{\Sigma}(\mathcal{A})$ played between a challenger and an adversary $\mathcal{A}$:*

1. *The challenger generates a key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and gives the public key pk to $\mathcal{A}$.*
2. *The adversary $\mathcal{A}$ may adaptively request signatures on tuples $(m, i, (r_j)_{j\neq i})$, where $i \in [n]$ is chosen by the adversary. For each query $(m, i, (r_j)_{j\neq i})$, the challenger randomly samples $r \leftarrow\!\!\$\ \mathcal{R}$ (the "uncorruptible" part of the randomness) and the remaining uncorrupted coins $r_i \leftarrow\!\!\$\ \mathcal{R}_i$, computes a signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m, (r, r_1, \ldots, r_m))$, and returns σ to $\mathcal{A}$.*
3. *Finally, the adversary outputs a forgery (m^*, σ^*). The experiment outputs 1 if:*
   - *$\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1$, and*
   - *no tuple $(m^*, i, (r_j)^*_{j\neq i})$ was previously queried by the adversary.*

   Otherwise, the experiment outputs 0.

The advantage of an adversary $\mathcal{A}$ in breaking the CEUFCMA *security of the scheme Σ is defined as:*

$$\mathsf{Adv}_{\Sigma}^{\mathsf{CEUFCMA}}(\mathcal{A}) = \Pr\left[\mathsf{Exp}_{\Sigma}^{\mathsf{CEUFCMA}}(\mathcal{A}) = 1\right]$$

where $\mathsf{Exp}_{\Sigma}^{\mathsf{CEUFCMA}}(\mathcal{A})$ is defined as above. A signature scheme Σ is said to be n-user CEUFCMA*-secure if the advantage $\mathsf{Adv}_{\Sigma}^{\mathsf{CEUFCMA}}(\mathcal{A})$ of any polynomial-time adversary $\mathcal{A}$ is negligible.*

It is noted that every signature scheme can trivially be converted into a corruptible $(n+1)$-source signature scheme as defined above by defining $\mathsf{Sign}^*(m, \mathsf{sk}; (r, r_1, \cdots, r_n)) := \mathsf{Sign}(m, \mathsf{sk}; r)$ (that is, ignoring the corruptible part of the randomness and using only the incorruptible part). However, not every $(n+1)$-source signature scheme is a $(n+1)$-source corruptible signature scheme. The results presented build upon the observation that specific constructions of $(n+1)$-source MPC-in-the-Head schemes are CEUFCMA secure. Looking ahead, r in the construction will correspond to the salt (which must be sampled randomly for each signature scheme but is publicly revealed by the signature), and each r_i will be a τ-tuple of seed roots for computing GGM trees.

7.5 An RSD-based threshold signature

The template presented in 7.3 is not an automated compiler; consequently, it does not produce a comprehensive abstract template capable of automatically deriving a threshold signature scheme from any MPC-in-the-Head scheme as input, along with a formal proof of security. This limitation arises due to the subtle low-level differences inherent in existing MPC-in-the-Head schemes: for instance, variations in how the GGM PPRF is instantiated, methods of injecting salt into PRG evaluations to mitigate the collision attack of [DN19], approaches to hashing the $\tau \cdot N$ commitments (either by hashing the full concatenation or by first hashing the τ blocks of N commitments individually before hashing the resulting τ hashes), and choices between computing the auxiliary string directly from the virtual party share or from the VOLE, among other details.
Each MPCitH scheme is also accompanied by its own security analysis. While many analyses broadly adhere to a similar template, differences in low-level details persist. Moreover, two primary approaches to security analysis are generally employed: most MPC-in-the-Head protocols follow the original Picnic analysis [ZCD+20], which is sometimes updated to model the GGM PPRF directly in the random oracle model to facilitate extraction [CHT23] or to model the GGM PPRF as a multi-instance PPRF to achieve tight security while relying only on AES for better efficiency [HJ24]. In parallel, VOLE-in-the-Head protocols typically utilize the proof methodology introduced by FAEST [LMØ+23], which significantly differs from the analyses of previous schemes. At a high level, while other MPC-in-the-Head schemes demonstrate standard soundness and are vulnerable to the Kales-Zaverucha attack [KZ20a] when the underlying identification scheme has more than three rounds (necessitating

parameter adjustments), the VOLE consistency check used in FAEST and subsequent works enables them to achieve the stronger notion of *round-by-round soundness* [CCH+19; CMS19], which, in particular, prevents security loss when compiling multi-round identification schemes via Fiat-Shamir.
Even if a similar analysis is expected (but not formally claimed) to apply to the majority of MPC-in-the-Head [KZ20b; BDK+21; ZCD+20; DDO+19; FJR22; AGH+23; BKP+23; ABC+23; ABB+23; BFR23; GSR23; DGO+23; CHT23; HJ24] and VOLE-in-the-Head [BBS+23; LMØ+23; ZLH24; BBM+24; OTX24] protocols, in this section only a detailed case analysis is presented: the signature scheme presented in Chapter 5.

7.5.1 The Threshold-Friendly variant of the signature scheme

Outlined below is a threshold-friendly variant of the signature scheme defined in Chapter 5, based on the regular syndrome decoding assumption. Beyond adapting the construction of Chapter 5 to align with the introduced recipe and make it "threshold-friendly," additional updates incorporate optimizations introduced more recently, such as the hypercube technique and other minor refinements. Consequently, the description might differ significantly from that of Chapter 5 while remaining the same signature scheme augmented with modern generic optimizations. In the following, $D = \log N$ denotes the logarithm of the number N of virtual parties.

The PPRF. The construction relies on a multi-instance puncturable pseudorandom function, as defined in Chapter 6. It is useful to recall that in this context, F denotes the GGM PPRF, instantiated with a length-doubling PRG, defined as $\mathsf{PRG}_{\mathsf{salt}}(x) = (\mathsf{AES}_{\mathsf{salt}_0}(x) \oplus x, \mathsf{AES}_{\mathsf{salt}_1}(x) \oplus x)$, where $\mathsf{salt}_0, \mathsf{salt}_1$ are two λ-bit random strings (parsed from the 2λ-bit random salt salt given as input to the multi-instance PPRF), and AES is modeled as an ideal cipher – this implies that the resulting PPRF is indeed (tightly) multi-instance secure as proved in Chapter 6–. Given a salt salt, a root seed sd, and an index i, $\mathsf{sd}_i \leftarrow \mathsf{F}_{\mathsf{salt}}(\mathsf{sd}, i)$ denotes the i-th leaf of the GGM tree computed using $\mathsf{PRG}_{\mathsf{salt}}$.
To simplify notation, given a tuple $\mathsf{cp} = (i^*, v^1, \cdots, v^D, \mathsf{sd}^)$, $\mathsf{F}_{\mathsf{salt}}(\mathsf{cp}, i)$ denotes the following procedure that computes the PPRF outputs from a co-path instead of a root seed:

- view $v^1, \cdots, v^D$ as the λ-bit seeds on the nodes of the co-path from the root to the i^*-th leaf (i.e., the blue nodes on Figure 7.2);
- if $i \neq i^*$, compute sd_i using $\mathsf{PRG}_{\mathsf{salt}}$ from its closer ancestor on the co-path (for example, if $i = 1$, that would be the seed v^2 on the leftmost blue node on Figure 7.2);
- else, if $i = i^*$, output $\mathsf{sd}_{i^*} = \mathsf{sd}^*$.

Eventually, given a salt salt, a root seed sd, and a leaf index i, $\mathsf{CoPath}_{\mathsf{salt}}(\mathsf{sd}, i)$ denotes the procedure which recomputes the entire GGM tree and outputs the D-tuple of seeds on the nodes on the co-path to the leaf i.

Key generation. In the proposed signature scheme, the key generation algorithm randomly samples a syndrome decoding instance (H, y) with solution x. The algorithm is described below.

KeyGen(1^λ)

Inputs: A security parameter λ.

1. Sample $\mathsf{sd} \leftarrow \{0,1\}^\lambda$ and set $H \leftarrow \mathsf{PRG}(\mathsf{sd})$ where $H = (H'|I) \in \mathbb{F}_2^{k \times K}$ is a parity-check matrix in systematic form.
2. Sample $(x|x_2) \leftarrow\!\$\ \mathsf{Reg}_w(\mathbb{F}_2^K)$ with $x \in \mathbb{F}_2^{K-k}$ and set $y \leftarrow H' \cdot x \oplus x_2$.
3. Divide x into n additive shares $\tilde{x}_i$ for $i \in [n]$;
4. Set $\mathsf{pk} = (\mathsf{sd}, y)$ and $\mathsf{sk} = (H, (\tilde{x}_1, \cdots, \tilde{x}_n), y)$.

Protocol 19: Key generation algorithm of the signature scheme

Signing. The signing algorithm is presented below.

Sign$(m, \mathsf{sk}; (r, r_1, \cdots, r_n))$

Inputs: A message $m \in \{0,1\}^{2\lambda}$ and secret key $\mathsf{sk} = (H, (\tilde{x}_1, \cdots, \tilde{x}_n), y)$.

Randomness:

- Parse r as $\mathsf{salt} \in \{0,1\}^{2\lambda}$ and derive $(\mathsf{salt}_1, \cdots, \mathsf{salt}_\tau) \leftarrow \mathsf{PRG}(\mathsf{salt})$.
- Parse each r_j as $(K^{e,j})_{e \le \tau}$ and further parse each $K^{e,j}$ as either $\mathsf{sd}^{e,j} \in \{0,1\}^\lambda$ or as $\mathsf{cp}^{e,j} = (i^*_{e,j}, v^1_{e,j}, \cdots, v^D_{e,j}, \mathsf{sd}^*_{e,j}) \in [N] \times (\{0,1\}^\lambda)^{D+1}$. // $\mathsf{cp}^{e,j}$ corresponds to using as randomness input a co-path to a leaf $i^*_{e,j}$ and a $i^*_{e,j}$-th leaf seed $\mathsf{sd}^*_{e,j}$ instead of a root seed $\mathsf{sd}^{e,j}$. Note that forcing a choice of co-path to a leaf $i^*_{e,j}$ yields a correct signature only if $i^e = i^*_{e,j}$ in Phase 4. This randomness input is never used in an honest use of the signing algorithm, but is allowed to specify the behavior of Sign given corrupted inputs.

Phase 1. For each iteration $e \in [\tau]$ and each $j \in [n]$:

- For $d = 1$ to D, set $(X^{e,j}_{d,0}, R^{e,j}_{d,0}, U^{e,j}_{d,0}) \leftarrow (0,0,0) \in \mathbb{F}_2^{K-k} \times \mathbb{F}_2^{K-k} \times \mathbb{F}_3^{K-k}$;
- Set $x^{e,j}_N \leftarrow \tilde{x}_j$, $u^e_N \leftarrow 0$, and $r^{e,j} \leftarrow 0$;
- **For** $i = 1$ **to** $N - 1$:
  1. Compute $\mathsf{sd}^{e,j}_i \leftarrow \mathsf{F}_{\mathsf{salt}}(\mathsf{sd}^{e,j}, i)$; // $\mathsf{F}_{\mathsf{salt}}(\mathsf{cp}^{e,j}, i)$ if $K^{e,j} = \mathsf{cp}^{e,j}$
  2. $(x^{e,j}_i, r^{e,j}_i, u^{e,j}_i, \mathsf{com}^{e,j}_i) \leftarrow \mathsf{PRG}(\mathsf{sd}^{e,j}_i)$;
     // $(x^{e,j}_i, r^{e,j}_i, u^{e,j}_i, \mathsf{com}^{e,j}_i) \in \mathbb{F}_2^{K-k} \times \mathbb{F}_2^{K-k} \times \mathbb{F}_3^{K-k} \times \{0,1\}^\lambda$.
  3. $x^{e,j}_N \leftarrow x^{e,j}_N \oplus x^{e,j}_i$, $r^{e,j} \leftarrow r^{e,j} \oplus r^{e,j}_i$ and $u^{e,j}_N \leftarrow u^{e,j}_N + u^{e,j}_i \bmod 3$;

4. Compute the virtual parties' views

$$(x_i^e, r_i^e, u_i^e) = \left(\bigoplus_j x_i^{e,j}, \bigoplus_j r_i^{e,j}, \sum_j u_i^{e,j} \bmod 3 \right).$$

- **On node** N: // computing the auxiliary string
  1. Compute $\mathsf{sd}_N^{e,j} \leftarrow \mathsf{F}_{\mathsf{salt}}(\mathsf{sd}^{e,j}, N)$ // $\mathsf{F}_{\mathsf{salt}}(\mathsf{cp}^{e,j}, N)$ if $K^{e,j} = \mathsf{cp}^{e,j}$
  2. Set $(r_N^{e,j}, \mathsf{com}_N^{e,j}) \leftarrow \mathsf{PRG}(\mathsf{sd}_N^{e,j})$;
  3. Set $r^{e,j} \leftarrow r^{e,j} \oplus r_N^{e,j}$, $r^e \leftarrow \bigoplus_j r^{e,j}$, $u^e \leftarrow \sum_{i=1}^{N-1} u_i^e \bmod 3$, and $u_N^e \leftarrow r^e - u^e \bmod 3$.
  4. Set the N-th virtual party's view

$$(x_N^e, r_N^e, u_N^e) = \left(\bigoplus_j x_N^{e,j}, \bigoplus_j r_N^{e,j}, u_N^e \bmod 3 \right)$$

  5. Define $\mathsf{aux}_N^e \leftarrow (x_N^e, u_N^e)$;
- **For** $i = 1$ **to** N: // hypercube aggregation
  - For all $d \leq D$ such that $i[d] = 0$, set: // $i[d]$ is the d-th bit of i.
    - $X_{d,0}^{e,j} \leftarrow X_{d,0}^{e,j} \oplus x_i^{e,j}$;
    - $R_{d,0}^{e,j} \leftarrow R_{d,0}^{e,j} \oplus r_i^{e,j}$;
    - $U_{d,0}^{e,j} \leftarrow U_{d,0}^{e,j} + u_i^{e,j} \bmod 3$;
  - Set the hypercube-aggregated views to

$$\left(X_{d,0}^e, R_{d,0}^e, U_{d,0}^e\right) = \left(\bigoplus_j X_{d,0}^{e,j}, \bigoplus_j R_{d,0}^{e,j}, \sum_j U_{d,0}^{e,j} \bmod 3 \right).$$

- Get $h_1^j \leftarrow \mathsf{H}_1(\mathsf{com}_1^{1,j}, \cdots, \mathsf{com}_N^{1,j}, \cdots, \mathsf{com}_1^{\tau,j}, \cdots, \mathsf{com}_N^{\tau,j})$; // Accumulate the commitments inside the hash rather than storing and hashing all at once.
- Set $h_1 \leftarrow \mathsf{H}_1(m, \mathsf{salt}, h_1^1, \cdots, h_1^n)$;

Phase 2.

1. $(\pi^e)_{e \leq \tau} \leftarrow \mathsf{PRG}_1(h_1)$. // $\pi^e \in \mathsf{Perm}([K-k])$.

Phase 3. For each iteration $e \in [\tau]$:

1. $z_1^e \leftarrow x \oplus \pi^e(r^e)$, $z_2^e \leftarrow H' \cdot z_1^e \oplus y$, and $z^e \leftarrow (z_1^e || z_2^e)$;
2. For $d = 1$ to D, set:

- $z^e_{d,0}[1] \leftarrow X^e_{d,0} \oplus \pi^e(R^e_{d,0})$, $z^e_{d,0}[2] \leftarrow H' \cdot z^e_{d,0}[1] \oplus y$, and $z^e_{d,0} \leftarrow (z^e_{d,0}[1]||z^e_{d,0}[2])$;
- $z^e_{d,1}[1] \leftarrow z^e_1 \oplus z^e_{d,0}$, $z^e_{d,1}[2] \leftarrow z^e_2 \oplus z^e_{d,0}[2]$, and $z^e_{d,1} \leftarrow (z^e_{d,1}[1]||z^e_{d,1}[2])$;
- $\bar{x}^e_{d,0} \leftarrow z^e + (1 - z^e) \cdot \pi^e(U^e_{d,0})$;
- $\bar{x}^e_{d,1} \leftarrow x - \bar{x}^e_{d,0} \bmod 3$.
- For $b = 0, 1$, set $\mathsf{msg}^e_{d,b} \leftarrow (z^e_{d,b}, \mathsf{BHW}_3(\bar{x}^e_{d,b}))$.

3. Get $h_2 \leftarrow \mathsf{H}_2(m, \mathsf{salt}, h_1, (\mathsf{msg}^e_{d,b})_{d \le D, b \in \{0,1\}, e \le \tau})$;

Phase 4.

- Set $(i^e)_{e \le \tau} \leftarrow \mathsf{PRG}_2(h_2)$. // $i^e \in [N]$.
- For $e = 1$ to τ, if there exists $j \in [n]$ such that $K^{e,j} = \mathsf{cp}^{e,j}$, denoting $i^*_{e,j}$ the first component of $\mathsf{cp}^{e,j}$, if $i^*_{e,j} \neq i^e$, raise a flag $\mathsf{flag} = \bot$. Else, raise a flag $\mathsf{flag} = \top$.
- For all e, j such that $K^{e,j} = \mathsf{cp}^{e,j} = (i^*_{e,j}, v^1_{e,j}, \cdots, v^D_{e,j}, \mathsf{sd}^*_{e,j})$, define $\mathsf{copath}^{e,j} = (v^1_{e,j}, \cdots, v^D_{e,j})$.
- For all e, j such that $K^{e,j} = \mathsf{sd}^{e,j}$, define $\mathsf{copath}^{e,j} = \mathsf{CoPath}_{\mathsf{salt}}(\mathsf{sd}^{e,j}, i^e)$.

Phase 5. Output

$$\sigma = \left(\mathsf{salt}, h_1, h_2, \left(\mathsf{copath}^{e,j}, \mathsf{com}^{e,j}_{i^e}\right)_{e \le \tau,\, j \le n}, (z^e, \mathsf{aux}^e_N)_{e \le \tau}\right), \mathsf{flag}.$$

Protocol 20: Signing algorithm of the signature scheme

Verification. The verification algorithm is presented below.

Verify(m, pk, σ)

Inputs. A public key $\mathsf{pk} = (H, y)$, a message $m \in \{0,1\}^*$, a signature σ.

1. Parse the signature as follows:

$$\sigma = \left(\mathsf{salt}, h_1, h_2, \left(\mathsf{copath}^{e,j}, \mathsf{com}^{e,j}_{i^e}\right)_{e \le \tau,\, j \le n}, (z^e, \mathsf{aux}^e_N)_{e \le \tau}\right)$$

2. Recompute $(\pi^e)_{e \le \tau} \leftarrow \mathsf{PRG}_1(h_1)$, where $\pi^e \in \mathsf{Perm}([K - k])$;
3. Recompute $(i^e)_{e \le \tau} \leftarrow \mathsf{PRG}_2(h_2)$ and parse each i^e as a D-bit string $(b^e_d)_{d \le D}$.
4. For each iteration $e \in [\tau]$,
   - For $d = 1$ to D:
     - Denote $b = 1 - b^e_d$;
     - Set $(X^e_{d,b}, R^e_{d,b}, U^e_{d,b}) \leftarrow (0,0,0) \in \mathbb{F}_2^{K-k} \times \mathbb{F}_2^{K-k} \times \mathbb{F}_3^{K-k}$;

- For each $i \neq i^e$:
  - Recompute $\mathsf{sd}_i^{e,j}$ from $\mathsf{copath}^{e,j}$ for each $j \in [n]$;
  - If $i \neq N$, recompute $(x_i^{e,j}, r_i^{e,j}, u_i^{e,j}, \mathsf{com}_i^e) \leftarrow \mathsf{PRG}(\mathsf{sd}_i^{e,j})$; else, parse aux_N^e as (x_N^e, u_N^e), and compute $r_N^e = \bigoplus_j r_N^{e,j}$ where $r_N^{e,j} \leftarrow \mathsf{PRG}(\mathsf{sd}_N^{e,j})$;
  - If $i[d] = b$, update:
    - $X_{d,b}^e \leftarrow X_{d,b}^e \oplus \bigoplus_j x_i^e$;
    - $R_{d,b}^e \leftarrow R_{d,b}^e \oplus \bigoplus_j r_i^e$;
    - $U_{d,b}^e \leftarrow U_{d,b}^e + \sum_j u_i^e \bmod 3$;
- Recompute $(\mathsf{msg}_{d,b}^e)_{d \leq D, b \in \{0,1\}, e \leq \tau}$ by simulating the Phase 3 of the signing algorithm as below:
  - $z_{d,0}^e \leftarrow \left(X_d^e \oplus \pi^e(R_{d,0}^e) || H' \cdot z_{d,0}^e[1] \oplus y\right)$;
  - $z_{d,1}^e \leftarrow \left(z_1^e \oplus z_{d,0}^e || z_2^e \oplus z_{d,0}^e[2]\right)$;
  - $\bar{x}_{d,b}^e \leftarrow z^e + (1 - z^e) \cdot \pi(U_{d,b}^e)$;
  - $\mathsf{msg}_{d,b}^e \leftarrow (z_{d,b}^e, \mathsf{BHW}_6(\bar{x}_{d,b}^e))$;
  - $\mathsf{msg}_{d,1-b}^e \leftarrow (z_{d,1-b}^e, 1 - \mathsf{BHW}_6(\bar{x}_{d,b}^e) \bmod 3)$

5. Check if $h_1 = \mathsf{H}_1(m, \mathsf{salt}, h_1^1, \cdots, h_1^n)$ where

$$h_1^j \leftarrow \mathsf{H}_1(\mathsf{com}_1^{1,j}, \cdots, \mathsf{com}_N^{1,j}, \cdots, \mathsf{com}_1^{\tau,j}, \cdots, \mathsf{com}_N^{\tau,j});$$

6. Check if $h_2 = \mathsf{H}_2(m, \mathsf{salt}, h_1, (\mathsf{msg}_{d,b}^e)_{d \leq D, b \in \{0,1\}, e \leq \tau})$;
7. Output 1 if both conditions are satisfied.

Protocol 21: Verification algorithm of the signature scheme

7.5.2 Security Analysis of the Threshold-Friendly Signature Scheme

In this section, it is proved that the signature scheme described on Protocol 19, Protocol 20 and Protocol 21 satisfies n-user corruptible existential unforgeability against chosen-message attacks 7.4.1. Note that since the adversary is not allowed any signature query in the EUFKO game (Definition 4.1.3), it is not needed to consider a corruptible variant of the notion.

7.5.2.1 Corruptible existential unforgeability

Theorem 7.5.1. *Assume that* F *is a* (q_s, τ)*-instance* $(t, \epsilon_{\mathsf{F}})$*-secure PPRF, that* PRG *is a* (q_s, τ)*-instance* $(t, \epsilon_{\mathsf{PRG}})$*-secure PRG, and that any adversary running in time* t *has at advantage at most* ϵ_{SD} *against the regular syndrome decoding problem. Model the hash functions* $\mathsf{H}_1, \mathsf{H}_2$ *as random oracles with output of length* 2λ*-bit and the pseudorandom generator* PRG_2 *as a random oracle. Then corrupted chosen-message adversary against the signature scheme*

described in Protocol 14, running in time t, making q_s signing queries, and making q_1, q_2, q_3 queries, respectively, to the random oracles $\mathsf{H}_1, \mathsf{H}_2$ and PRG_2, succeeds in outputting a valid forgery with probability

$$\mathsf{Adv}^{\mathsf{CEUFCMA}}_{\Sigma}(\mathcal{A}) \leq \frac{q_s\,(q_s + q_1 + q_2 + q_3)}{2^{2\lambda}} + \varepsilon_{\mathsf{F}} + \varepsilon_{\mathsf{PRG}} + \epsilon_{\mathsf{SD}} + \Pr[X + Y = \tau] + \frac{1}{2^{\lambda}},$$

where $X = \max_{\alpha \in Q_1}\{X_\alpha\}$ and $Y = \max_{\beta \in Q_2}\{Y_\beta\}$ with $X_\alpha \sim \mathsf{Binomial}(\tau, \mathsf{p})$ and $Y_\beta \sim \mathsf{Binomial}\left(\tau - X, \frac{1}{N}\right)$ where Q_1 and Q_2 are sets of all queries to oracles H_1 and H_2 and p is a statistical failure event bounded in Lemma 5.3.1, and set to 2^{-132} in parameter choices.

Proof. The security analysis proceeds in two parts: first bounding $\mathsf{Adv}^{\mathsf{EUFKO}}(\mathcal{A})$, then bounding $\mathsf{Adv}^{\mathsf{CEUFCMA}}(\mathcal{A})$ using $\mathsf{Adv}^{\mathsf{EUFKO}}(\mathcal{A})$. The first half of the analysis, bounding $\mathsf{Adv}^{\mathsf{EUFKO}}(\mathcal{A})$, is identical to the analysis in section 6.4.2 (up to replacing the single GGM tree by n GGM trees whose leaves are summed everywhere in the analysis) and yields

$$\mathsf{Adv}^{\mathsf{EUFKO}}_{\Sigma}(\mathcal{A}) \leq \epsilon_{\mathsf{SD}} + \Pr[X + Y = \tau] + \frac{1}{2^{\lambda}},$$

where $\varepsilon_{\mathsf{SD}}, X, Y$ are as as defined in the statement of Theorem 7.5.1. The crux of the analysis in this context lies in reducing corruptible existential unforgeability against chosen message attacks to EUFKO security. This is captured by the following lemma:

Lemma 7.5.1 (EUFKO $\implies$ CEUFCMA)**.**

$$\mathsf{Adv}^{\mathsf{CEUFCMA}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{EUFKO}}(\mathcal{A}) + \frac{q_s\,(q_s + q_1 + q_2 + q_3)}{2^{2\lambda}} + \epsilon_{\mathsf{F}} + \epsilon_{\mathsf{PRG}}$$

To prove Lemma 7.5.1, a sequence of experiments involving $\mathcal{A}$ is defined, where the first corresponds to the experiment in which $\mathcal{A}$ interacts with the real signature scheme, and the last one is an experiment in which $\mathcal{A}$ is using only random element independent from the witness.

Game 1 (Gm1). This corresponds to the actual interaction of $\mathcal{A}$ with the real signature scheme. Upon receiving a query $(m, j, (r_j)_{i \neq j})$ from $\mathcal{A}$, the signing oracle samples $r_j := (\mathsf{sd}^{e,j})_{e \leq \tau} \leftarrow\!\$\ (\{0,1\}^{\lambda})^{\tau}$, and return $\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}; (r_1, \cdots, r_n))$. Denote $\mathsf{Gm}\ell(\mathsf{Forge})$, the event that after interacting with the corruptible signing oracle in Game ℓ, $\mathcal{A}$ generates a valid signature σ^* for a message m^* that was not previously queried to the signing oracle.

Game 2 (Gm2). In this game, abort if the sampled salt salt collides with the value sampled in any of the previous queries to the hash functions H_1 or H_2, or if the input to PRG_2 collides with the value obtained in any of the previous queries. Therefore, this probability can be bounded by

$$|\Pr[\mathsf{Gm1}(\mathsf{Forge})] - \Pr[\mathsf{Gm2}(\mathsf{Forge})]| \leq \frac{q_s \cdot (q_s + q_1 + q_2 + q_3)}{2^{2\lambda}}$$

Game 3 (Gm3). The difference with the previous game is that now before signing a message uniformly random values h_1, h_2 and $(i^e)_{e\leq\tau}$ are chosen. Since Phase 1, Phase 3 and Phase 5 are computed as before and the only change compared to the previous game is that the output of H_1 is randomly set as h_1, the output of H_2 as h_2 and the output of $\mathsf{PRG}_2(h_2)$ as $(i^e)_{e\leq\tau}$. A difference in the forgery probability can only happen in the event that a query to H_1, H_2 or PRG_2 was made before; however, in this scenario, Game 2 aborts. Therefore,

$$\Pr[\mathsf{Gm2}(\mathsf{Forge})] = \Pr[\mathsf{Gm3}(\mathsf{Forge})].$$

Game 4 (Gm4) in this game, upon receiving a query $(m, j, (r_i)_{i\neq j})$ from $\mathcal{A}$, after sampling $r_j := (\mathsf{sd}^{e,j})_{e\leq\tau} \leftarrow\!\!\$\ (\{0,1\}^\lambda)^\tau$, compute the leaf seed $\mathsf{sd}^{e,j}_{i^e} = \mathsf{F}_{\mathsf{salt}}(\mathsf{sd}^{e,j}, i^e)$ and the corresponding co-path $\mathsf{copath}^{e,j} = \mathsf{CoPath}_{\mathsf{salt}}(\mathsf{sd}^{e,j}, i^e)$. Set $\mathsf{cp}^{e,j} = (i^e, \mathsf{copath}^{e,j}, \mathsf{sd}^{e,j}_{i^e})$ for $e = 1$ to τ and $r_j = (\mathsf{cp}^{e,j})_{u\leq\tau}$. Run $\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}; (r_1, \cdots, r_n))$.
This game is a purely syntactic change: by the design of the signing algorithm, it will output exactly the same signature as in Game 3 (observe that since PRG_2 is programmed to output $(i^e)_e$, no aborts of the signing algorithm will be triggered). Therefore, it holds that

$$\Pr[\mathsf{Gm3}(\mathsf{Forge})] = \Pr[\mathsf{Gm4}(\mathsf{Forge})].$$

Game 5 (Gm5) in this game, upon receiving a query $(m, j, (r_i)_{i\neq j})$ from $\mathcal{A}$, instead of sampling $r_j := (\mathsf{sd}^{e,j})_{e\leq\tau} \leftarrow\!\!\$\ (\{0,1\}^\lambda)^\tau$, sample for $e = 1$ to τ a uniformly random leaf seed $\mathsf{sd}^{e,j}_{i^e}$ and a uniformly random corresponding co-path $\mathsf{copath}^{e,j}$ (note that from Game 3, the indices i^e are sampled ahead of time). Set $\mathsf{cp}^{e,j} = (i^e, \mathsf{copath}^{e,j}, \mathsf{sd}^{e,j}_{i^e})$ for $e = 1$ to τ and $r_j = (\mathsf{cp}^{e,j})_{u\leq\tau}$. Run $\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}; (r_1, \cdots, r_n))$.
The only difference between Game 3 and Game 4 is that the leaf seed $\mathsf{sd}^{e,j}_{i^e}$ and its corresponding co-path $\mathsf{copath}^{e,j}$ are sampled uniformly at random. Distinguishing between the two games reduces therefore immediately to breaking the (Q,τ)-instance strong security of the PPRF, where Q is a bound on the number of signing queries from $\mathcal{A}$ (cf 6.2.2). Therefore, it holds that

$$|\Pr[\mathsf{Gm5}(\mathsf{Forge})] - \Pr[\mathsf{Gm4}(\mathsf{Forge})]| \leq \mathsf{Adv}^{\mathsf{F}}(\mathcal{A}) = \varepsilon_{\mathsf{F}}.$$

Game 6 (Gm6). In this game, upon receiving a query $(m, j, (r_i)_{i\neq j})$ from $\mathcal{A}$ and for $e = 1$ to τ, $(x^{e,j}_{i^e}, r^{e,j}_{i^e}, u^{e,j}_{i^e}, \mathsf{com}^{e,j}_{i^e}) \leftarrow\!\!\$\ \mathbb{F}_2^{K-k} \times \mathbb{F}_2^{K-k} \times \mathbb{F}_3^{K-k} \times \{0,1\}^\lambda$ are sampled at random. As the seed $\mathsf{sd}^{e,j}_{i^e}$ is uniformly random (independent of $\mathsf{copath}^{e,j}$ and never revealed by the signature (it is only used in Game 5 to construct $(x^{e,j}_{i^e}, r^{e,j}_{i^e}, u^{e,j}_{i^e}, \mathsf{com}^{e,j}_{i^e}) \leftarrow \mathsf{PRG}(\mathsf{sd}^{e,j}_{i^e})$), this game is indistinguishable from the previous one by a direct application of the multi-instance security of PRG (Section 6.2.1), and it holds that

$$|\Pr[\mathsf{Gm5}(\mathsf{Forge})] - \Pr[\mathsf{Gm4}(\mathsf{Forge})]| \leq \mathsf{Adv}^{\mathsf{PRG}}(\mathcal{A}) = \epsilon_{\mathsf{PRG}}.$$

Game 7 (Gm7). In this game, the behavior of the emulation in Phase 3 is modified. Namely, the messages $(\mathsf{msg}^e_{d,b})_{d\leq D, b\in\{0,1\}, e\leq\tau}$ are constructed using instead the same procedure as

the verifier in step 4 of the Verify algorithm (Protocol 21). Note that this process yields an identical construction of the $\mathsf{msg}^e_{d,b}$ by the correctness of the verification algorithm, and can be used in the emulation because the challenges i^e are sampled ahead of time. Note also that the witness x is at this stage only used in two places: in the computation of x^e_N in aux^e_N, and in the computation of $z^e_1 = x \oplus \pi^e(r^e)$. It holds that

$$\Pr[\mathsf{Gm7}(\mathsf{Forge})] = \Pr[\mathsf{Gm6}(\mathsf{Forge})].$$

Game 8 (Gm8). In this game, $\mathsf{aux}^e_N \leftarrow\!\$\ \mathbb{F}_2^{K-k} \times \mathbb{F}_3^{K-k}$ and $z^e_1 \leftarrow\!\$\ \mathbb{F}_3^{K-k}$ for $e = 1$ to τ are sampled. Note that $\mathsf{aux}^e_N = (x^e_N, u^e_N)$ is constructed as

$$x^e_N = x \oplus \bigoplus_{i=1}^{N} \bigoplus_{j=1}^{n-1} x_i^{e,j}, \qquad u^e_N = r^e - \sum_{i=1}^{N-1} \sum_{j=1}^{n} u_i^{e,j} \bmod 3.$$

Because each of these terms is masked by a uniformly random value (respectively $x^{e,j}_{i^e}$ and $u^{e,j}_{i^e}$), and because $x \oplus \pi^e(r^e)$ is masked by $\pi^e(r^{e,j}_{i^e})$, all are uniformly random over $\mathbb{F}_2^{K-k}$, $\mathbb{F}_3^{K-k}$, and $\mathbb{F}_2^{K-k}$ respectively, and it holds that

$$\Pr[\mathsf{Gm8}(\mathsf{Forge})] = \Pr[\mathsf{Gm7}(\mathsf{Forge})].$$

Observe that in Game 8, the emulation does not use the witness x anymore, hence it does not need the secret key sk. Therefore, an adversary outputting a forgery in Game 8 immediately implies an adversary with the same success probability against the EUFKO security of the signature scheme:

$$\Pr[\mathsf{Gm8}(\mathsf{Forge})] = \mathsf{Adv}^{\mathsf{EUFKO}}(\mathcal{A}).$$

This concludes the proof of Lemma 7.5.1 and Theorem 7.5.1. □

7.5.3 The functionality $\mathcal{F}_{2,3}$

On Functionality 1, the mod2-to-mod3 functionality is represented. A "corruptible" variant of the functionality is described, where the corrupted parties define their output shares, and the honest parties' output shares are sampled consistently with the shares of the corrupted parties.

$\mathcal{F}_{2,3}$

Parameters. The functionality interacts with n users $\mathcal{U}_1, \cdots, \mathcal{U}_n$. $\mathcal{C}$ denotes the (possibly empty) subset of corrupted users.

Input. Each user $\mathcal{U}_j$ sends a vector $u_j \in \mathbb{F}_2^t$. Additionally, each corrupted user $\mathcal{U}_j$ sends a vector $u'_j \in \mathbb{F}_3^t$. The functionality aborts if it receives incorrectly formatted inputs, or if they do not all have the same length.

Functionality.

- Compute $u = \bigoplus_{j \le n} u_j$.
- Compute $u' = \sum_{j \in \mathcal{C}} u'_j$.
- Sample $n - |\mathcal{C}|$ uniformly random shares $(u'_j)_{j \in [n] \setminus \mathcal{C}}$ of $u - u' \bmod 3$ over $\mathbb{F}_3$.
- Output u'_j to each honest user $\mathcal{U}_j$.

Functionality 1: ideal functionality for converting sums from mod2 *to* mod3

7.5.3.1 Instantiating the functionality In this section, simple protocols are provided to securely instantiate $\mathcal{F}_{2,3}$ in the malicious setting using recent results on pseudorandom correlation generators [BCG+18; BCG+19b; BCG+19a]. Before proceeding with their description, two observations are made:

- The simulation of the threshold signing protocol is oblivious to whether the output of $\mathcal{F}_{2,3}$ is correct or not. This is because Sim emulates the protocol without using the share output by $\mathcal{F}_{2,3}$ (or uses a random share instead when $\mathsf{flag} = \mathsf{fail}$), and because the protocol does not verify that the corrupted parties are actually using the correct output obtained via the functionality. This is a relatively standard behavior in threshold signatures: the correct behavior of the participants is guaranteed by the validity of the signature produced by the protocol. In fact, many existing threshold signatures leverage a similar observation to obtain more efficient constructions. In this context, this implies that only the instantiation of the functionality is required to guarantee correctness when the users are honest and *privacy of the honest parties' inputs* when some participants are malicious. This eliminates the need for any expensive zero-knowledge proofs or similar mechanisms.
- In the model where key generation is executed by a trusted entity (typically the owner of sk), it can be assumed that the trusted dealer additionally generates and includes in the key shares sk_j *helper strings* (e.g., correlated randomness) that the parties can use to facilitate the secure instantiation of $\mathcal{F}_{2,3}$.

A construction for $n = 2$. Equipped with these observations, the constructions are now presented, starting with the two-party setting ($n = 2$). First, some background on PCGs is recalled. A PCG for a target correlation C (a 2-party correlation is a distribution over pairs of values) is a pair $(\mathsf{Setup}, \mathsf{Expand})$ such that

- $\mathsf{Setup}(1^\lambda)$ produces short keys c_0, c_1, and
- $\mathsf{Expand}(\sigma, c_\sigma)$ outputs a long string y_σ,

such that (y_0, y_1) are indistinguishable from a random sample from C. A formal definition is provided in Section 3.4.6 (taken almost verbatim from [BCC+23]). The (length-t) OLE

correlation over a field $\mathbb{F}$ refers to the following correlation: the two users $\mathcal{U}_1, \mathcal{U}_2$ receive $(\mathbf{z}_1, \mathbf{x}) \in (\mathbb{F}^t)^2$ and $(\mathbf{z}_2, \mathbf{y}) \in (\mathbb{F}^t)^2$ respectively, such that

$$\mathbf{z}_0 + \mathbf{z}_1 = \mathbf{x} \odot \mathbf{y},$$

where $\odot$ denotes the Schur (i.e., component-wise) product. A PCG for the OLE correlation is said to be *programmable* if, informally, the randomness used to generate $\mathbf{x}$ or $\mathbf{y}$ can be fixed across different instances (such that $\mathcal{U}_1$ can obtain an OLE $(\mathbf{z}_1, \mathbf{x})$ with a user $\mathcal{U}_2$, and a second tuple $(\mathbf{z}'_1, \mathbf{x})$ with the same $\mathbf{x}$ with another user $\mathcal{U}_3$). A formal definition is provided in Section 3.4.6.
The following result from [BCC+23; BBC+24] is used:

Lemma 7.5.2. *Assuming the quasi-abelian syndrome decoding assumption, a programmable PCG exists for the length-t OLE correlation over $\mathbb{F}_q$ for any $q > 2$. Furthermore, the key size is bounded by $O(\lambda^3 \log t)$, and the runtime of* Expand *is $\tilde{O}(t)$.*

The work of [BBC+24] is noted to introduce a number of algorithmic and low-level optimizations that demonstrate that this PCG achieves very good concrete performance over small fields: [BBC+24] reports generating 12 million OLEs over $\mathbb{F}_4$. While the setting here is slightly different as it operates over $\mathbb{F}_3$, most of their optimizations carry over directly to this setting, and a similar, if not better, efficiency is expected.

$\Pi^2_{2,3}$

Parameters. A PCG (Setup, Expand) for length-t OLEs over $\mathbb{F}_3$

Trusted setup. The dealer generates $(c_1, c_2) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ and adds them to the secret keys of $\mathcal{U}_1, \mathcal{U}_2$ respectively.

Protocol.

1. For $i = 1, 2$, given inputs $\mathbf{r}_i \in \mathbb{F}_2^t$, user $\mathcal{U}_i$ generates $(\mathbf{z}_i, \mathbf{x}_i) \leftarrow \mathsf{Expand}(i, c_i)$ and broadasts $\mathbf{v}_i = \mathbf{x}_i + \mathbf{r}_i \bmod 3$.
2. User $\mathcal{U}_1$ outputs $\mathbf{z}_1 - \mathbf{v}_2 \odot \mathbf{x}_1 + \mathbf{r}_1 \bmod 3$.
3. User $\mathcal{U}_2$ outputs $\mathbf{z}_2 + \mathbf{v}_1 \odot \mathbf{r}_2 + \mathbf{r}_2 \bmod 3$.

Protocol 22: A 2-party protocol for securely instantiating the functionality $\mathcal{F}_{2,3}$

It is first shown that Protocol 22 is correct. Over $\mathbb{F}_3$, the following holds:

$$\begin{aligned}
&(\mathbf{z}_1 - \mathbf{v}_2 \odot \mathbf{x}_1 + \mathbf{r}_1) + (\mathbf{z}_2 + \mathbf{v}_1 \odot \mathbf{r}_2 + \mathbf{r}_2) \\
=&(\mathbf{z}_1 + \mathbf{z}_2) + (\mathbf{r}_1 + \mathbf{x}_1) \odot \mathbf{r}_2 - (\mathbf{r}_2 + \mathbf{x}_2) \odot \mathbf{x}_1 + \mathbf{r}_1 + \mathbf{r}_2 \\
=&\ \mathbf{x}_1 \odot \mathbf{x}_2 + \mathbf{r}_1 \odot \mathbf{r}_2 + \mathbf{x}_1 \odot \mathbf{r}_2 - \mathbf{r}_2 \odot \mathbf{x}_1 - \mathbf{x}_1 \odot \mathbf{x}_2 + \mathbf{r}_1 + \mathbf{r}_2 \\
=&\ \mathbf{r}_1 \odot \mathbf{r}_2 + \mathbf{r}_1 + \mathbf{r}_2 \\
=&\ \mathbf{r}_1 \oplus \mathbf{r}_2.
\end{aligned}$$

Security immediately follows from the fact that, by the PCG security, $\mathbf{x}_i$ is computationally indistinguishable from a random vector over $\mathbb{F}_3^t$ given c_{3-i}. Consequently, $\mathbf{v}_i$ computationally masks $\mathbf{r}_i$ over $\mathbb{F}_3$.

General case. For an arbitrary number of users, a generalization of protocol $\Pi_{2,3}$ in a tree-based fashion is relied upon. The description begins with a protocol $\Pi^{2^i}_{\mathsf{xor}}$, where 2^i parties, given as input additive shares over $\mathbb{F}_3$ of two bits a, b, generate additive shares over $\mathbb{F}_3$ of the xor $a \oplus b$.

$\Pi^{2^i}_{\mathsf{xor}}$

Trusted setup. The dealer samples $(u, v) \leftarrow\!\!\$\ \mathbb{F}_3^2$ and set $(u_j, v_j, w_j)_{j \le 2^i}$ to be random shares of $(u, v, u \cdot v)$ over $\mathbb{F}_3$. Each user $\mathcal{U}_j$ receives (u_j, v_j, w_j).

Protocol. The users have $\mathbb{F}_3$-shares $(a_j)_{j \le 2^i}$ and $(b_j)_{j \le 2^i}$ of bits a, b.

1. Each user $\mathcal{U}_j$ broadcasts $u_j + a_j \bmod 3$ and $v_j + b_j \bmod 3$. All users reconstruct $\sum_j (u_j + a_j) = u + a$ and $\sum_j (v_j + b_j) = v + b$.
2. Each user $\mathcal{U}_j$ outputs $z_j = (a + u)b_j - (v + b)u_j + w_j + a_j + b_j \bmod 3$.

Protocol 23: A 2^i-party protocol for computing shares of the XOR of two bits shared over $\mathbb{F}_3$

Correctness follows easily by inspection, as $\sum_j z_j = (a + u)b - (v + b)u + uv + a + b = ab + a + b = a \oplus b$ over $\mathbb{F}_3$, and security follows from the fact that each a_j (resp. b_j) is perfectly masked by u_j (resp. v_j) over $\mathbb{F}_3$. Equipped with protocol $\Pi^{2^i}_{\mathsf{xor}}$, a protocol is described (in the $\mathcal{F}^{2^i}_{\mathsf{xor}}$-hybrid model) that securely instantiates $\mathcal{F}_{2,3}$ with n users.

$\Pi^n_{2,3}$

Input. The parties $\mathcal{U}_j$ each have an input $r_j \in \mathbb{F}_2$.

Protocol. Assume that $n = 2^d$ is a power of 2. Place the n on the leaves of a full binary tree of depth d. The protocol proceeds in d rounds. In round i, all users who share a common ancestor on the $i + 1$-th layer interact. Set $i = 1$ to be the leaf layer.

1. Every 2^i-tuples S of users who have an ancestor on the layer $i+1$ (there are 2^{d-i-1} disjoint tuples) partition S into two equal-sized sets S_0, S_1 of users that share a common ancestor on the i-th layer. The users $(\mathcal{U}_j)_{j\in S_b}$ have additive shares of a bit r_{S_b} over $\mathbb{F}_3$. By adding dummy 0 shares, it is possible to assume that all users $(\mathcal{U}_j)_{j\in S}$ have 2^j-user shares of (r_{S_0}, r_{S_1}).
2. The users $(\mathcal{U}_j)_{j\in S}$ invoke $\mathcal{F}_{\mathsf{xor}}^{2^i}$ to obtain shares of $r_S := r_{S_0} \oplus r_{S_1}$ and set $i = i+1$
3. When $i = d$, all users output their $\mathbb{F}_3$-shares of $r_{[2^d]} = \bigoplus r_j$.

Protocol 24: An n-party protocol for securely instantiating the functionality $\mathcal{F}_{2,3}^n$ in the $\mathcal{F}_{\mathsf{xor}}^{2^i}$-hybrid model

The proof of correctness and privacy against malicious users of Protocol 24, in the $\mathcal{F}_{\mathsf{xor}}^{2^i}$-hybrid, is straightforward. The protocol $\Pi_{2,3}^n$ can be generalized to producing $\mathbb{F}_3$-shares of the XOR of $\mathbb{F}_2^t$-vectors via direct parallel repetition.
Eventually, instantiating $\mathcal{F}_{\mathsf{xor}}^{2^i}$ via $\Pi_{\mathsf{xor}}^{2^i}$ requires access to a trusted source of random 2^i-user Beaver triples over $\mathbb{F}_3$. This can be instantiated efficiently using the PCG for OLEs over $\mathbb{F}_3$ from [BCC+23]: as their PCG is programmable, and as explained in their work, if each pair of users is given a pair of (programmed) PCG seeds, they can locally recombine the pseudorandom outputs into 2^i-user $\mathbb{F}_3$-Beaver triples. This yields the following efficient instantiation of $\mathcal{F}_{2,3}$: in the trusted setup phase, the trusted dealer generates as many pairwise PCG seeds as required to instantiate $\Pi_{\mathsf{xor}}^{2^i}$ at every level of the tree-based construction from Protocol 24. Then, the parties locally expand their seeds into arbitrary length Beaver triples over $\mathbb{F}_3$ and run the efficient protocol from Protocol 24 to convert $\mathbb{F}_2$-shares into $\mathbb{F}_3$-shares.

7.5.4 The Threshold Signature

In this section, a threshold signature scheme constructed from the threshold-friendly signature scheme introduced in Section 7.5.1 is presented. The construction assumes the following building blocks and functionalities:

- Let Commit denote an extractable and equivocable commitment scheme. Concretely, a possible instantiation of Commit is as $\mathsf{Commit}(m;r) = \mathsf{H}'(m;r)$, where H' is a random oracle to which the simulator is given programmable access.
- $\mathcal{F}_{\mathsf{salt}}$ is a random coin-sampling functionality: upon receiving the signing session ID from all users, it samples $\mathsf{salt} \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$ and outputs it to all users. This functionality can be instantiated via a simple commit-and-open protocol.
- $\mathcal{F}_{2,3}$ denotes the mod2-to-mod3" functionality, which converts additive shares of a vector modulo 2 into additive shares of the same vector modulo 3. The ideal functionality is represented on Functionality 1. In Section 7.5.3.1, efficient protocols for

securely instantiating the functionality using pseudorandom correlation generators are introduced.

7.5.4.1 The threshold signing protocol

The whole protocol is described below.

Threshold Signing

Key-generation (trusted setup).

Inputs: A security parameter λ.

1. Run $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda)$.
2. Parse $\mathsf{sk} = (H, (\tilde{x}_1, \cdots, \tilde{x}_n)$. Output $\mathsf{sk}_j = (H, \tilde{x}_j, y)$ to each $\mathcal{U}_j$.

Sampling the salt. All parties invoke $\mathcal{F}_{\mathsf{salt}}$ and receive a global salt $\mathsf{salt} \in \{0,1\}^{2\lambda}$. All parties derive $(\mathsf{salt}_1, \cdots, \mathsf{salt}_\tau) \leftarrow \mathsf{PRG}(\mathsf{salt})$.

Building the trees. For each iteration $e \in [\tau]$, each user $\mathcal{U}_j$ proceeds as follows:

- Sample $\mathsf{sd}^{e,j} \leftarrow\!\$\ \{0,1\}^\lambda$;
- For $d = 1$ to D, set $(X^{e,j}_{d,0}, R^{e,j}_{d,0}, U^{e,j}_{d,0}) \leftarrow (0,0,0)$;
- Set $x^{e,j}_N \leftarrow \tilde{x}_j$, $u^e_N \leftarrow 0$, and $r^{e,j} \leftarrow 0$;
- **For** $i = 1$ **to** $N-1$:
  1. Compute $\mathsf{sd}^{e,j}_i \leftarrow \mathsf{F}_{\mathsf{salt}}(\mathsf{sd}^{e,j}, i)$;
  2. $(x^{e,j}_i, r^{e,j}_i, u^{e,j}_i, \mathsf{com}^{e,j}_i) \leftarrow \mathsf{PRG}(\mathsf{sd}^{e,j}_i)$;
  3. $x^{e,j}_N \leftarrow x^{e,j}_N \oplus x^{e,j}_i$, $r^{e,j} \leftarrow r^{e,j} \oplus r^{e,j}_i$ and $u^{e,j}_N \leftarrow u^{e,j}_N + u^{e,j}_i \bmod 3$;
- **On node** N**:**
  1. Compute $\mathsf{sd}^{e,j}_N \leftarrow \mathsf{F}_{\mathsf{salt}}(\mathsf{sd}^{e,j}, N)$
  2. Set $(r^{e,j}_N, \mathsf{com}^{e,j}_N) \leftarrow \mathsf{PRG}(\mathsf{sd}^{e,j}_N)$
  3. Set $r^{e,j} \leftarrow r^{e,j} \oplus r^{e,j}_N$
  4. Send $r^{e,j}$ to the $\mathcal{F}_{2,3}$ functionality in order to obtain $u^{e,j}$. // This is the **Shift** phase of the template protocol in Protocol 18.
  5. Set $u^{e,j}_N \leftarrow u^{e,j} - u^{e,j}_N \bmod 3$
  6. Broadcast $(x^{e,j}_N, u^{e,j}_N)$. Upon receiving all shares, all user reconstruct $\mathsf{aux}^e_N = (x^e_N, u^e_N) = \left(\bigoplus_j x^{e,j}_N, \sum_j u^{e,j}_N \bmod 3\right)$;
- For $i = 1$ to N, for all $d \le D$ such that $i[d] = 0$, set: // $i[d]$ is the d-th bit of the integer i.
  - $X^{e,j}_{d,0} \leftarrow X^{e,j}_{d,0} \oplus x^{e,j}_i$;
  - $R^{e,j}_{d,0} \leftarrow R^{e,j}_{d,0} \oplus r^{e,j}_i$;
  - $U^{e,j}_{d,0} \leftarrow U^{e,j}_{d,0} + u^{e,j}_i \bmod 3$.

Commit-and-open phase. For each iteration $e \in [\tau]$, each user $\mathcal{U}_j$ proceeds as follows:

- Set $h_1^j \leftarrow \mathsf{H}_1(\mathsf{com}_1^{1,j}, \cdots, \mathsf{com}_N^{1,j}, \cdots, \mathsf{com}_1^{\tau,j}, \cdots, \mathsf{com}_N^{\tau,j})$.
- Broadcast $c^j \leftarrow\!\!\$\ \mathsf{Commit}(h^j)$.
- Upon receiving all $(c^\ell)_{\ell\in[n]}$, broadcast h^j and the opening of c^j.
- If an opening does not verify, abort the protocol. Else, all users compute $h_1 \leftarrow \mathsf{H}_1(m, \mathsf{salt}, h_1^1, \cdots, h_1^n)$.
- Compute $(\pi^e)_{e\leq\tau} \leftarrow \mathsf{PRG}_1(h_1)$. // The use of a permutation to shuffle the correlated randomness replaces the VOLE consistency check in 5.2.1.

Distributed virtual protocol. Let $(1_j^p)_{j\in[n]}$ denote arbitrary shares of 1 modulo $p = 2$ or 3.

- Each user $\mathcal{U}_j$ sets $z_1^{e,j} \leftarrow \tilde{x} \oplus \pi^e(r^{e,j})$, $z_2^{e,j} \leftarrow H' \cdot z_1^{e,j} \oplus y \cdot 1_j^2$, and $z^{e,j} \leftarrow (z_1^{e,j} || z_2^e)$.
- All users broadcast $z^{e,j}$ and reconstruct $z^e = \bigoplus_j z^{e,j}$.
- For $d = 1$ to D, set:
  - $z_{d,0}^{e,j}[1] \leftarrow X_{d,0}^{e,j} \oplus \pi^e(R_{d,0}^{e,j})$, $z_{d,0}^{e,j}[2] \leftarrow H' \cdot z_{d,0}^{e,j}[1] \oplus y$, and $z_{d,0}^{e,j} \leftarrow (z_{d,0}^{e,j}[1] || z_{d,0}^{e,j}[2])$;
  - $z_{d,1}^{e,j}[1] \leftarrow z_1^{e,j} \oplus z_{d,0}^{e,j}$, $z_{d,1}^{e,j}[2] \leftarrow z_2^{e,j} \oplus z_{d,0}^{e,j}[2]$, and $z_{d,1}^{e,j} \leftarrow (z_{d,1}^{e,j}[1] || z_{d,1}^{e,j}[2])$;
  - $\bar{x}_{d,0}^{e,j} \leftarrow z^e \cdot 1_j^3 + (1 - z^e) \cdot \pi^e(U_{d,0}^{e,j}) \bmod 3$
  - $\bar{x}_{d,1}^e \leftarrow \tilde{x}_j - \bar{x}_{d,0}^{e,j} \bmod 3$.
  - For $b = 0, 1$, set $\mathsf{msg}_{d,b}^{e,j} \leftarrow (z_{d,b}^{e,j}, \mathsf{BHW}_6(\bar{x}_{d,b}^{e,j}))$.
  - For $b = 0, 1$, all users broadcast $\mathsf{msg}_{d,b}^{e,j}$ and reconstruct $\mathsf{msg}_{d,b}^e = \sum_j \mathsf{msg}_{d,b}^{e,j} \bmod 3$.
- Set $h_2 \leftarrow \mathsf{H}_2(m, \mathsf{salt}, h_1, (\mathsf{msg}_{d,b}^e)_{d\leq D, b\in\{0,1\}, e\leq\tau})$.
- Set $(i^e)_{e\leq\tau} \leftarrow \mathsf{PRG}_2(h_2)$.
- Each user $\mathcal{U}_j$ broadcasts $\mathsf{copath}^{e,j} = \mathsf{CoPath}_{\mathsf{salt}}(\mathsf{sd}^{e,j}, i^e)$ and $\mathsf{com}_{i_e}^{e,j}$ for $e = 1$ to τ. // This is the **Opening** phase in the template protocol of Protocol 18.

Output. The parties abort if the signature σ below does not verify.

$$\sigma = \left(\mathsf{salt}, h_1, h_2, \left(\mathsf{copath}^{e,j}, \mathsf{com}_{i_e}^{e,j}\right)_{e\leq\tau,\, j\leq n}, (z^e, \mathsf{aux}_N^e)_{e\leq\tau}\right).$$

7.5.4.2 Security analysis

Theorem 7.5.2. *Let $\mathcal{A}$ denote an adversary corrupting at most $n-1$ users, engaging in any polynomial number Ns of threshold signing sessions, making at most Q queries to the random oracle H_1 and to the ideal cipher, and outputting a forgery (m^*, σ^*) after all sessions, where m^* is a message that was never signed during a signing session. Then in the $(\mathcal{F}_{\mathsf{salt}}, \mathcal{F}_{2,3})$-hybrid*

model, it holds that

$$\Pr[\mathsf{Verify}(m, \mathsf{pk}, \sigma^*) = 1] \leq \frac{\mathsf{Q}^2 + \mathsf{N}s\mathsf{n}\mathsf{Q}}{2^{2\lambda}} + \mathsf{Adv}^{\mathsf{Commit}}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{CEUFCMA}}(\mathcal{A}).$$

Proof. Let $\mathcal{A}$ denote an adversary corrupting all users except $\mathcal{U}_\ell$ (the case of a smaller number of corrupted parties proceeds similarly to the all-but-one corruption case). Below is a description of a simulator Sim that interacts with the corruptible signing functionality and is given (non-programming) access to the random oracle H_1 and to the ideal cipher underlying the multi-instance PPRF.

KeyGen. Sim invokes the key generation of the corruptible signing oracle and receives $\mathsf{pk} = (\mathsf{sd}, y)$. It recomputes $H \leftarrow \mathsf{PRG}(\mathsf{sd})$ and picks $(\tilde{x}_j)_{j\neq\ell} \leftarrow\!\$\ (\mathbb{F}_2^{K-k})^{n-1}$ uniformly at random. It outputs $\mathsf{sk}_j = (H, \tilde{x}_j, y)$ to each corrupted user $\mathcal{U}_j$.

Sampling the salt. Sim honestly emulates $\mathcal{F}_{\mathsf{salt}}$.

Building the trees. Sim stores the inputs $\tilde{r}^{e,j}$ of all corrupted users to $\mathcal{F}_{2,3}$ and randomly samples $(u^{e,j})_{j\neq\ell} \leftarrow\!\$\ (\mathbb{F}_3^{K-k})^{n-1}$. It returns $u^{e,j}$ to each corrupted user $\mathcal{U}_j$.

Commit-and-open phase. Sim broadcasts a dummy commitment c^ℓ. Upon receiving all commitments $(c^j)_{j\neq\ell}$, for each $j \neq \ell$,

- Sim extracts the value $\tilde{h}^j$ contained in c^j.
- Sim searches among all queries to H_1 for a preimage of $\tilde{h}^j$ of the form $\mathbf{com}^j = (\mathsf{com}_1^{1,j}, \cdots, \mathsf{com}_N^{1,j}, \cdots, \mathsf{com}_1^{\tau,j}, \cdots, \mathsf{com}_N^{\tau,j})$. If there is no such preimage, or if there are multiple preimages, or if the preimage is not correctly formatted, Sim raises a flag fail.
- For all tuples $\mathbf{com}^j$ for which Sim did not raise a flag fail, for $e = 1$ to τ, Sim searches among all queries to the ideal cipher for preimages on the nodes of a GGM tree with salt salt_e and commitments $(\mathsf{com}_1^{e,j}, \cdots, \mathsf{com}_N^{e,j})$ at the leaves. If neither a root seed $\mathsf{sd}^{j,e}$ nor a set of seeds $(v_{e,j}^1, \cdots, v_{e,j}^D)$ on the co-path to a leaf $i_{e,j}^*$ together with the $i_{e,j}^*$-th leaf seed $\mathsf{sd}_{e,j}^*$ is found, it raises a flag fail. Otherwise, it sets $K^{e,j}$ to be either $\mathsf{sd}^{e,j}$ or $\mathsf{cp}^{e,j} = (i_{e,j}^*, v_{e,j}^1, \cdots, v_{e,j}^D, \mathsf{sd}_{e,j}^*)$ and $r^j \leftarrow (K^{e,j})_{e\leq\tau}$.
- If Sim did not raise a flag fail, it sends $(m, \ell, (r^j)_{j\neq\ell})$ to the corruptible signing functionality. Upon receiving a signature σ, it recomputes h_1^ℓ from the signature via the verification algorithm.
- Otherwise, Sim picks random root seeds $\mathsf{sd}^{e,\ell}$ and constructs h_1^ℓ as an honest user, using $u^{e,\ell} \leftarrow\!\$\ \mathbb{F}_3^{K-k}$ as simulated output of $\mathcal{F}_{2,3}$.
- Sim adapts the opening of c^ℓ to h_1^ℓ.

Distributed virtual protocol (flag $\neq$ fail)**.** Provided that no flag fail is raised, Sim can recompute at this stage from the r^j all values $z^{e,j}$, $\mathsf{msg}_{d,b}^{e,j}$ for $j \neq \ell$, $e = 1$ to τ, $d = 1$ to D, and $b \in \{0,1\}$. From σ, by running the verification algorithm, Sim obtains z^e and $\mathsf{msg}_{d,b}^{e}$ for $e = 1$ to τ, $d = 1$ to D, and $b \in \{0,1\}$.

- For $e = 1$ to τ, $z^{e,\ell} \leftarrow z^e \oplus \bigoplus_{j\neq\ell} z^{e,j}$ is defined and broadcasted.
- For $e = 1$ to τ, $d = 1$ to D, and $b \in \{0,1\}$, $\mathsf{msg}_{d,b}^{e,\ell} \leftarrow \mathsf{msg}_{d,b}^{e} - \sum_{j\neq\ell} \mathsf{msg}_{d,b}^{e,j} \bmod 3$ is broadcasted.
- Eventually, $(\mathsf{copath}^{e,\ell}, \mathsf{com}_{i^e}^{e,\ell})$ for $e = 1$ to τ is computed from σ and broadcasted.

Distributed virtual protocol (flag = fail)**.** If flag = fail, uniformly random $z^{e,\ell}$, $\mathsf{msg}_{d,b}^{e,\ell}$ are broadcasted, as well as the correct co-path and commitment $(\mathsf{copath}^{e,\ell}, \mathsf{com}_{i^e}^{e,\ell})$ computed from random root seeds.

As Sim is only given access to the corruptible signing functionality, the advantage of $\mathcal{A}$ in outputting a valid forgery (m^*, σ^*) after interacting polynomially many times with Sim is at most $\mathsf{Adv}^{\mathsf{CEUFCMA}}(\mathcal{A})$. It is now shown that Sim's emulation is indistinguishable from a real interaction with $\mathcal{U}_\ell$. This is proved via a sequence of game hops.

Game 1 (Gm1). This is the real game. The game honestly runs the key generation and distributes sk_j to each corrupted user $\mathcal{U}_j$. The game honestly emulates $\mathcal{U}_\ell$.

Game 2 (Gm2). In this game, a flag fail is raised if any collision occurs in H_1 or in the ideal cipher. As the total number of queries to either H_1 or the ideal cipher is at most Q, $\Pr[\mathsf{fail}(\mathsf{Gm2})] \leq \frac{\mathsf{Q}^2}{2^{2\lambda}}$ holds.

Game 3 (Gm3). In this game, a flag fail is raised if a signature contains $h_1^j = \mathsf{H}_1(\mathbf{com}^j)$ such that $\mathbf{com}^j$ was not queried to H_1 before sending c^j, and a flag ver is raised if any of the two checks of this same signature verifies. By the one-wayness of H_1, $\Pr[\mathsf{ver} \mid \mathsf{fail}(\mathsf{Gm3})] \leq \frac{\mathsf{N}s\mathsf{n}\mathsf{Q}}{2^{2\lambda}}$ holds, where the bound is reached only if all signing sessions are performed in parallel and the adversary attempts to find a preimage to any c^j of any signing session after sending c^j (and before running the distributed virtual protocol).

Game 4 (Gm4). In this game, when flag = fail, $\mathcal{U}_\ell$ is emulated as the simulator Sim, computing an honest GGM tree but sampling dummy $u^{e,\ell}$, $z^{e,\ell}$, $\mathsf{msg}_{d,b}^{e,\ell}$. Conditioned on flag = fail, with probability at least $1 - \frac{\mathsf{N}s\mathsf{n}\mathsf{Q}}{2^{2\lambda}}$, both checks (for h_1 and h_2) fail due to an incorrect h_1^j. In this case, all honestly computed $z^{e,\ell}$, $\mathsf{msg}_{d,b}^{e,\ell}$ are distributed as random elements over $\mathbb{F}_2^K$ and $\mathbb{F}_3^K \times \mathbb{F}_3^w$, respectively (they are random shares of values for which one share is missing).

Game 5 (Gm5). This game is the simulation, where Sim does not use the secret key sk. Instead, Sim interacts with the corruptible signing oracle. Observe that whenever flag $\neq$ fail, the only difference between the transcript of the simulated interaction and the transcript of Game 4 is the commitment c^ℓ, which Sim computes as an equivocal commitment to a dummy string. Therefore, the advantage of any adversary $\mathcal{A}$ in distinguishing Game 4 from Game 5 is at most $\mathsf{Adv}^{\mathsf{Commit}}(\mathcal{A})$. This concludes the proof of Theorem 7.5.2. □

Chapter 8

Conclusion and Open Questions

8.1 Conclusion

The research carried out in this thesis brings notable advancements in code-based post-quantum signature schemes, with a strong focus on integrating the MPC-in-the-Head (MPCitH) paradigm to achieve secure, efficient, and scalable digital signatures: the thesis introduces new code-based primitives, with related meaningful optimizations and extends them to a practical threshold signature construction.

The first contribution presents a five-round zero-knowledge proof of knowledge built on the Regular Syndrome Decoding (RSD) problem using the MPCitH paradigm. By applying the Fiat-Shamir heuristic, the interactive protocol is turned into a digital signature scheme with competitive performance in terms of both signature size and computational efficiency.

The second contribution builds on the first by integrating an innovative hypercube technique that simplifies and accelerates verification, delivering a faster signature generation process. Moreover, the contribution focuses on enhancing the MPCitH paradigm by introducing multi-instance puncturable pseudorandom functions (PPRFs). Replacing conventional hash-based approaches with block ciphers drastically reduces both signing and verification times, pushing the performance boundaries of MPCitH signatures.

The third contribution generalizes the single-user signature scheme into a threshold signature framework, solving key challenges associated with distributed signing across multiple parties. The resulting construction maintains scalability, balancing the signature size and the number of participants, and demonstrates real-world applicability for decentralized systems.

In summary, this thesis combines rigorous theoretical design with practical performance improvements, aligning with NIST's post-quantum cryptographic objectives while addressing practical needs in blockchain and other decentralized applications.

8.2 Future Directions

While the results presented in this manuscript mark a significant step in optimizing code-based digital signatures and extending them to multi-user schemes, a number of open questions remain and future directions can expand across different horizons:

1. Enhancing the efficiency of threshold signatures and extending them to multisignatures, to enable their integration into blockchain systems;
2. Addressing fundamental technical challenges in the underlying MPC-in-the-Head framework;
3. Integrate combinatorial approaches with modern techniques, to further optimize signature schemes.

Towards Multisignature Schemes

Extending threshold signatures into full multi-signature schemes is a natural next step. While threshold signatures allow a specific subset of users to sign a message collaboratively, multi-signature schemes go further: any arbitrary group of users can produce a single, compact signature. However, before achieving such a huge level of scalability, some improvements in the threshold protocol have to be studied.

Signature Size Optimization Current constructions achieve a signature size of $\lambda^2 \cdot N$, but the lower bounds established by [DKR24] suggest that $O(\lambda \cdot N)$ is theoretically possible for a (black-box) threshold signature scheme from MPC-in-the-head.

Question 8.2.1. *Is it possible to construct a threshold MPCitH signature with size $O(\lambda \cdot N)$, achieving better scalability?*

Round Complexity Reduction The current protocol requires $O(\log N)$ rounds. Reducing this to a constant would enhance usability in real-world settings where network delays are a factor.

Question 8.2.2. *Can the round complexity of threshold MPCitH be reduced to a constant while maintaining its efficiency and security guarantees?*

Optimizing MPC-in-the-Head

As largely discussed, extending MPC-in-the-Head to collaborative multi-user settings like threshold and multisignatures creates non-trivial communication and computational overhead. Defining new techniques for thresholdizing MPCitH and improving its performance is crucial for practical uses.

Non-Blackbox Optimization The lower bound of [DKR24] shows that no black-box solution will have a constant signature size. Unfortunately, non-blackbox evaluation seems to require distributively computing the GGM trees, which incurs prohibitive costs.

Question 8.2.3. *Could it be possible to design specially crafted MPCitH signatures that can be efficiently "thresholdized" via a non-blackbox distributed protocol that does not require fully expanding a distributed GGM tree?*

Leveraging the Half-Tree Methodology The half-tree technique [GYW+23] is a recent methodology that can be used to reduce the number of calls to the ideal cipher in GGM-style PPRFs. Adapting this methodology could further optimize MPCitH-based constructions, with a corresponding enhancement in their threshold variants.

Question 8.2.4. *Can the half-tree methodology be adapted to yield multi-instance PPRFs in the ideal cipher model?*

Combining Methodologies

The combinatorial approaches to RSD-based signatures developed in this thesis could be enhanced by integrating them with modern methodologies like VOLEitH. Such hybrid approaches may further reduce signature size and improve overall efficiency.

Question 8.2.5. *How can combinatorial RSD-based techniques be integrated with VOLEitH to achieve compact and efficient signature schemes?*

By addressing these interconnected challenges, future work can strengthen the theoretical framework and enhance the practical applicability of the presented cryptographic primitives.

Bibliography

Bibliography

[AF04] Masayuki Abe and Serge Fehr. "Adaptively Secure Feldman VSS and Applications to Universally-Composable Threshold Cryptography". In: *CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. LNCS. Springer, Heidelberg, Aug. 2004, pp. 317–334. DOI: 10.1007/978-3-540-28628-8_20.

[ABB+23] Gora Adj, Stefano Barbero, Emanuele Bellini, Andre Esser, Luis Rivera-Zamarripa, Carlo Sanna, Javier Verbel, and Floyd Zweydinger. *MiRitH: Efficient Post-Quantum Signatures from MinRank in the Head*. Cryptology ePrint Archive, Paper 2023/1666. 2023. URL: https://eprint.iacr.org/2023/1666.

[01] *Advanced Encryption Standard (AES)*. National Institute of Standards and Technology, NIST FIPS PUB 197, U.S. Department of Commerce. Nov. 2001.

[AGH+23] Carlos Aguilar Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. "The Return of the SDitH". In: *EUROCRYPT 2023, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. LNCS. Springer, Heidelberg, Apr. 2023, pp. 564–596. DOI: 10.1007/978-3-031-30589-4_20.

[ABC+23] Nicolas Aragon, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, and Matthieu Rivain. "MIRA: a Digital Signature Scheme based on the MinRank problem and the MPC-in-the-Head paradigm". In: *arXiv preprint arXiv:2307.08575* (2023).

[AFS03] Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. *A Fast Provably Secure Cryptographic Hash Function*. Cryptology ePrint Archive, Report 2003/230. https://eprint.iacr.org/2003/230. 2003.

[Bab85] László Babai. "Trading Group Theory for Randomness". In: *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*. 1985, pp. 421–429. DOI: 10.1145/22145.22192.

[BLS+24] Renas Bacho, Julian Loss, Gilad Stern, and Benedikt Wagner. "HARTS: High-threshold, Adaptively Secure, and Robust Threshold Schnorr Signatures". In: *Cryptology ePrint Archive* 2024 (2024). https://eprint.iacr.org/2024/280, Paper ID TBD.

[BLT+24] Renas Bacho, Julian Loss, Stefano Tessaro, Benedikt Wagner, and Chenzhi Zhu. "Twinkle: Threshold Signatures from DDH with Full Adaptive Security". In: *Advances in Cryptology – EUROCRYPT 2024*. Lecture Notes in Computer Science. Available at https://eprint.iacr.org/2023/1482. Springer, 2024.

[BBM+24] Carsten Baum, Ward Beullens, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. "One Tree to Rule Them All: Optimizing GGM Trees and OWFs for Post-Quantum Signatures". In: *ASIACRYPT 2024 (to appear)*. 2024. URL: https://eprint.iacr.org/2024/490.

[BBS+23] Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. *Publicly Verifiable Zero-Knowledge and Post-Quantum Signatures From VOLE-in-the-Head*. Cryptology ePrint Archive, Paper 2023/996. 2023. URL: https://eprint.iacr.org/2023/996.

[BDK+21] Carsten Baum, Cyprien Delpech de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. "Banquet: Short and Fast Signatures from AES". In: *PKC 2021, Part I*. Ed. by Juan Garay. Vol. 12710. LNCS. Springer, Heidelberg, May 2021, pp. 266–297. DOI: 10.1007/978-3-030-75245-3_11.

[BJM+12] 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: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Heidelberg, Apr. 2012, pp. 520–536. DOI: 10.1007/978-3-642-29011-4_31.

[BCK+22] Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. "Better than Advertised Security for Non-interactive Threshold Signatures". In: *CRYPTO 2022, Part IV*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13510. LNCS. Springer, Heidelberg, Aug. 2022, pp. 517–550. DOI: 10.1007/978-3-031-15985-5_18.

[BG93] Mihir Bellare and Oded Goldreich. "On Defining Proofs of Knowledge". In: *CRYPTO'92*. Ed. by Ernest F. Brickell. Vol. 740. LNCS. Springer, Heidelberg, Aug. 1993, pp. 390–420. DOI: 10.1007/3-540-48071-4_28.

[BR93] Mihir Bellare and Phillip Rogaway. "Random oracles are practical: a paradigm for designing efficient protocols". In: CCS '93. Fairfax, Virginia, USA: Association for Computing Machinery, 1993, pp. 62–73. ISBN: 0897916298. DOI: 10.1145/168588.168596. URL: https://doi.org/10.1145/168588.168596.

[BR94] Mihir Bellare and Phillip Rogaway. "Entity Authentication and Key Distribution". In: *CRYPTO'93*. Ed. by Douglas R. Stinson. Vol. 773. LNCS. Springer, Heidelberg, Aug. 1994, pp. 232–249. DOI: 10.1007/3-540-48329-2_21.

[BR96] Mihir Bellare and Phillip Rogaway. "The Exact Security of Digital Signatures-How to Sign with RSA and Rabin". In: *Advances in Cryptology — EUROCRYPT '96*. Ed. by Ueli Maurer. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 399–416. ISBN: 978-3-540-68339-1.

[BFR23] Ryad Benadjila, Thibauld Feneuil, and Matthieu Rivain. *MQ on my Mind: Post-Quantum Signatures from the Non-Structured Multivariate Quadratic Problem*. Cryptology ePrint Archive, Paper 2023/1719. 2023. URL: https://eprint.iacr.org/2023/1719.

[Ber10] Daniel J. Bernstein. "Grover vs. McEliece". In: *The Third International Workshop on Post-Quantum Cryptography, PQCRYPTO 2010*. Ed. by Nicolas Sendrier. Springer, Heidelberg, May 2010, pp. 73–80. DOI: 10.1007/978-3-642-12929-2_6.

[BLN+09] Daniel J. Bernstein, Tanja Lange, Ruben Niederhagen, Christiane Peters, and Peter Schwabe. "FSBday". In: *INDOCRYPT 2009*. Ed. by Bimal K. Roy and Nicolas Sendrier. Vol. 5922. LNCS. Springer, Heidelberg, Dec. 2009, pp. 18–38.

[BLP11] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. "Smaller Decoding Exponents: Ball-Collision Decoding". In: *CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. LNCS. Springer, Heidelberg, Aug. 2011, pp. 743–760. DOI: 10.1007/978-3-642-22792-9_42.

[BLP+11] Daniel J. Bernstein, Tanja Lange, Christiane Peters, and Peter Schwabe. "Really Fast Syndrome-Based Hashing". In: *AFRICACRYPT 11*. Ed. by Abderrahmane Nitaj and David Pointcheval. Vol. 6737. LNCS. Springer, Heidelberg, July 2011, pp. 134–152.

[BKP+23] Luk Bettale, Delaram Kahrobaei, Ludovic Perret, and Javier Verbel. *Biscuit: New MPCitH Signature Scheme from Structured Multivariate Polynomials*. Cryptology ePrint Archive, Paper 2023/1760. 2023. URL: https://eprint.iacr.org/2023/1760.

[BGK+22] Loic Bidoux, Philippe Gaborit, Mukul Kulkarni, and Victor Mateu. "Code-based Signatures from New Proofs of Knowledge for the Syndrome Decoding Problem". In: *arXiv preprint arXiv:2201.05403* (2022).

[BS91] Eli Biham and Adi Shamir. "Differential Cryptanalysis of DES-like Cryptosystems". In: *Journal of Cryptology* 4.1 (1991), pp. 3–72. DOI: 10.1007/BF00630563.

[Blu82] Manuel Blum. "Coin Flipping by Telephone". In: *Proc. IEEE Spring COMPCOM*. 1982, pp. 133–137.

[BM82] Manuel Blum and Silvio Micali. "How to generate cryptographically strong sequences of pseudo random bits". In: *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*. 1982, pp. 112–117. DOI: 10.1109/SFCS.1982.72.

[BBC+24] Maxime Bombar, Dung Bui, Geoffroy Couteau, Alain Couvreur, Clément Ducros, and Sacha Servan-Schreiber. "FOLEAGE: $\mathbb{F}_4$OLE-Based Multi-Party Computation for Boolean Circuits". In: *ASIACRYPT 2024 (to appear)*. 2024. URL: https://eprint.iacr.org/2024/429.

[BCC+23] Maxime Bombar, Geoffroy Couteau, Alain Couvreur, and Clément Ducros. "Correlated Pseudorandomness from the Hardness of Quasi-Abelian Decoding". In: *CRYPTO 2023, Part IV*. LNCS. Springer, Heidelberg, Aug. 2023, pp. 567–601. DOI: 10.1007/978-3-031-38551-3_18.

[BW13] Dan Boneh and Brent Waters. "Constrained Pseudorandom Functions and Their Applications". In: *ASIACRYPT 2013, Part II*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8270. LNCS. Springer, Heidelberg, Dec. 2013, pp. 280–300. DOI: 10.1007/978-3-642-42045-0_15.

[BHO+12] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes". In: *2012 IEEE Symposium on Security and Privacy*. 2012, pp. 553–567. DOI: 10.1109/SP.2012.44.

[BCG+21] Elette Boyle, Nishanth Chandran, Niv Gilboa, Divya Gupta, Yuval Ishai, Nishant Kumar, and Mayank Rathee. "Function Secret Sharing for Mixed-Mode and Fixed-Point Secure Computation". In: *EUROCRYPT 2021, Part II*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12697. LNCS. Springer, Heidelberg, Oct. 2021, pp. 871–900. DOI: 10.1007/978-3-030-77886-6_30.

[BCG+18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. "Compressing Vector OLE". In: *ACM CCS 2018*. Ed. by David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang. ACM Press, Oct. 2018, pp. 896–912. DOI: 10.1145/3243734.3243868.

[BCG+22] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. "Correlated Pseudorandomness from Expand-Accumulate Codes". In: *CRYPTO 2022, Part II*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13508. LNCS. Springer, Heidelberg, Aug. 2022, pp. 603–633. DOI: 10.1007/978-3-031-15979-4_21.

[BCG+19a] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. "Efficient Two-Round OT Extension and Silent Non-Interactive Secure Computation". In: *ACM CCS 2019*. Ed. by Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz. ACM Press, Nov. 2019, pp. 291–308. DOI: 10.1145/3319535.3354255.

[BCG+19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. "Efficient Pseudorandom Correlation Generators: Silent OT Extension and More". In: *CRYPTO 2019, Part III*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11694. LNCS. Springer, Heidelberg, Aug. 2019, pp. 489–518. DOI: 10.1007/978-3-030-26954-8_16.

[BCG+20a] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. "Correlated Pseudorandom Functions from Variable-Density LPN". In: *61st FOCS*. IEEE Computer Society Press, Nov. 2020, pp. 1069–1080. DOI: 10.1109/FOCS46700.2020.00103.

[BCG+20b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. "Efficient Pseudorandom Correlation Generators from Ring-LPN". In: *CRYPTO 2020, Part II*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Heidelberg, Aug. 2020, pp. 387–416. DOI: 10.1007/978-3-030-56880-1_14.

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. "Function Secret Sharing". In: *EUROCRYPT 2015, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. LNCS. Springer, Heidelberg, Apr. 2015, pp. 337–367. DOI: 10.1007/978-3-662-46803-6_12.

[BGI16a] Elette Boyle, Niv Gilboa, and Yuval Ishai. "Breaking the Circuit Size Barrier for Secure Computation Under DDH". In: *CRYPTO 2016, Part I*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9814. LNCS. Springer, Heidelberg, Aug. 2016, pp. 509–539. DOI: 10.1007/978-3-662-53018-4_19.

[BGI16b] Elette Boyle, Niv Gilboa, and Yuval Ishai. "Function Secret Sharing: Improvements and Extensions". In: *ACM CCS 2016*. Ed. by Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi. ACM Press, Oct. 2016, pp. 1292–1303. DOI: 10.1145/2976749.2978429.

[BGI19] Elette Boyle, Niv Gilboa, and Yuval Ishai. "Secure Computation with Preprocessing via Function Secret Sharing". In: *TCC 2019, Part I*. Ed. by Dennis Hofheinz and Alon Rosen. Vol. 11891. LNCS. Springer, Heidelberg, Dec. 2019, pp. 341–371. DOI: 10.1007/978-3-030-36030-6_14.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. "Functional Signatures and Pseudorandom Functions". In: *PKC 2014*. Ed. by Hugo Krawczyk. Vol. 8383. LNCS. Springer, Heidelberg, Mar. 2014, pp. 501–519. DOI: 10.1007/978-3-642-54631-0_29.

[BCC+24] Dung Bui, Eliana Carozza, Geoffroy Couteau, Dahmun Goudarzi, and Antoine Joux. "Short Signatures from Regular Syndrome Decoding, Revisited". In: *ASIACRYPT 2024 (to appear)*. 2024. URL: https://eprint.iacr.org/2024/252.

[BCS24] Dung Bui, Kelong Cong, and Cyprien Delpech de Saint Guilhem. *Improved All-but-One Vector Commitment with Applications to Post-Quantum Signatures*. Cryptology ePrint Archive, Paper 2024/097. https://eprint.iacr.org/2024/097. 2024. URL: https://eprint.iacr.org/2024/097.

[CKY09] Jan Camenisch, Aggelos Kiayias, and Moti Yung. "On the Portability of Generalized Schnorr Proofs". In: https://eprint.iacr.org/2009/050. 2009. URL: https://eprint.iacr.org/2009/050.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. "Fiat-Shamir: from practice to theory". In: *51st ACM STOC*. Ed. by Moses Charikar and Edith Cohen. ACM Press, June 2019, pp. 1082–1090. DOI: 10.1145/3313276.3316380.

[CGG+20] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. "UC Non-Interactive, Proactive, Threshold ECDSA with Identifiable Aborts". In: *ACM CCS 2020*. Ed. by Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna. ACM Press, Nov. 2020, pp. 1769–1787. DOI: 10.1145/3372297.3423367.

[CGJ+99] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. "Adaptive Security for Threshold Cryptosystems". In: *CRYPTO'99*. Ed. by Michael J. Wiener. Vol. 1666. LNCS. Springer, Heidelberg, Aug. 1999, pp. 98–115. DOI: 10.1007/3-540-48405-1_7.

[CC24] Eliana Carozza and Geoffroy Couteau. *On Threshold Signatures from MPC-in-the-Head*. Cryptology ePrint Archive, Paper 2024/1897. 2024. URL: https://eprint.iacr.org/2024/1897.

[CCJ23] Eliana Carozza, Geoffroy Couteau, and Antoine Joux. "Short Signatures from Regular Syndrome Decoding in the Head". In: *EUROCRYPT 2023, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. LNCS. Springer, Heidelberg, Apr. 2023, pp. 532–563. DOI: 10.1007/978-3-031-30589-4_19.

[CDM+22] Kevin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. "Statistical Decoding 2.0: Reducing Decoding to LPN". In: *ASIACRYPT 2022, Part IV*. Ed. by Shweta Agrawal and Dongdai Lin. Vol. 13794. LNCS. Springer, Heidelberg, Dec. 2022, pp. 477–507. DOI: 10.1007/978-3-031-22972-5_17.

[CHT23] Kevin Carrier, Valérian Hatey, and Jean-Pierre Tillich. *SDitH: A Code-Based Post-Quantum Digital Signature Scheme*. Tech. rep. Additional Round 1 Submission. National Institute of Standards and Technology, 2023. URL: https://sdith.org/.

[CS14] Shan Chen and John P. Steinberger. "Tight Security Bounds for Key-Alternating Ciphers". In: *EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer, Heidelberg, May 2014, pp. 327–350. DOI: 10.1007/978-3-642-55220-5_19.

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. "Succinct Arguments in the Quantum Random Oracle Model". In: *TCC 2019, Part II*. Ed. by Dennis Hofheinz and Alon Rosen. Vol. 11892. LNCS. Springer, Heidelberg, Dec. 2019, pp. 1–29. DOI: 10.1007/978-3-030-36033-7_1.

[CJ04] Jean-Sébastien Coron and Antoine Joux. *Cryptanalysis of a Provably Secure Cryptographic Hash Function*. Cryptology ePrint Archive, Report 2004/013. https://eprint.iacr.org/2004/013. 2004.

[CRR21] Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. "Silver: Silent VOLE and Oblivious Transfer from Hardness of Decoding Structured LDPC Codes". In: *CRYPTO 2021, Part III*. Ed. by Tal Malkin and Chris Peikert. Vol. 12827. LNCS. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 502–534. DOI: 10.1007/978-3-030-84252-9_17.

[CDX+16] Ronald Cramer, Ivan Damgard, Chaoping Xing, and Chen Yuan. "Amortized Complexity of Zero-Knowledge Proofs Revisited: Achieving Linear Soundness Slack". In: (2016). URL: https://eprint.iacr.org/2016/681.

[CD01] Ronald Cramer and Ivan Damgård. "Secure Distributed Linear Algebra in a Constant Number of Rounds". In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Heidelberg, Aug. 2001, pp. 119–136. DOI: 10.1007/3-540-44647-8_7.

[CKM23] Elizabeth C. Crites, Chelsea Komlo, and Mary Maller. "Fully Adaptive Schnorr Threshold Signatures". In: *CRYPTO 2023, Part I*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14081. Lecture Notes in Computer Science. Springer, Cham, Aug. 2023, pp. 678–709.

[DFG+14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. "Square Span Programs with Applications to Succinct NIZK Arguments". In: *ASIACRYPT 2014, Part I*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. LNCS. Springer, Heidelberg, Dec. 2014, pp. 532–550. DOI: 10.1007/978-3-662-45611-8_28.

[DR24] Sourav Das and Ling Ren. "Adaptively Secure BLS Threshold Signatures from DDH and co-CDH". In: *CRYPTO 2024, Part VII*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14926. Lecture Notes in Computer Science. Springer, Cham, Aug. 2024, pp. 251–284.

[Deb19] Thomas Debris-Alazard. "Cryptographie fondée sur les codes: nouvelles approches pour constructions et preuves; contribution en cryptanalyse". PhD thesis. Sorbonne université, 2019.

[DGO+23] Thomas Debris-Alazard, Philippe Gaborit, Ayoub Otmani, and Jean-Pierre Tillich. *RYDE: A Code-Based Post-Quantum Digital Signature Scheme*. Tech. rep. Additional Round 1 Submission. National Institute of Standards and Technology, 2023. URL: https://pqc-ryde.org/.

[DDO+19] Cyprien Delpech de Saint Guilhem, Lauren De Meyer, Emmanuela Orsini, and Nigel P. Smart. "BBQ: Using AES in Picnic Signatures". In: *SAC 2019*. Ed. by Kenneth G. Paterson and Douglas Stebila. Vol. 11959. LNCS. Springer, Heidelberg, Aug. 2019, pp. 669–692. DOI: 10.1007/978-3-030-38471-5_27.

[DH76] Whitfield Diffie and Martin E. Hellman. "New Directions in Cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638.

[DN19] Itai Dinur and Niv Nadler. "Multi-target Attacks on the Picnic Signature Scheme and Related Protocols". In: *EUROCRYPT 2019, Part III*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11478. LNCS. Springer, Heidelberg, May 2019, pp. 699–727. DOI: 10.1007/978-3-030-17659-4_24.

[DKR24] Jack Doerner, Yashvanth Kondi, and Leah Namisa Rosenbloom. "Sometimes You Can't Distribute Random-Oracle-Based Proofs". In: *Advances in Cryptology – CRYPTO 2024*. Ed. by Leonid Reyzin and Douglas Stebila. Cham: Springer Nature Switzerland, 2024, pp. 323–358. ISBN: 978-3-031-68388-6.

[EGK+20] Daniel Escudero, Satrajit Ghosh, Marcel Keller, Rahul Rachuri, and Peter Scholl. "Improved Primitives for MPC over Mixed Arithmetic-Binary Circuits". In: *CRYPTO 2020, Part II*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Heidelberg, Aug. 2020, pp. 823–852. DOI: 10.1007/978-3-030-56880-1_29.

[EKT24] Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. "Two-Round Threshold Signature from Algebraic One-More Learning with Errors". In: *CRYPTO 2024, Part VII*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14926. Lecture Notes in Computer Science. Springer, Cham, Aug. 2024, pp. 387–424.

[ES23] Andre Esser and Paolo Santini. *Not Just Regular Decoding: Asymptotics and Improvements of Regular Syndrome Decoding Attacks*. Cryptology ePrint Archive, Paper 2023/1568. 2023. URL: https://eprint.iacr.org/2023/1568.

[FJR21] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. *Shared Permutation for Syndrome Decoding: New Zero-Knowledge Protocol and Code-Based Signature*. Cryptology ePrint Archive, Report 2021/1576. https://eprint.iacr.org/2021/1576. 2021.

[FJR22] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. "Syndrome Decoding in the Head: Shorter Signatures from Zero-Knowledge Proofs". In: *CRYPTO 2022, Part II*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13508. LNCS. Springer, Heidelberg, Aug. 2022, pp. 541–572. DOI: 10.1007/978-3-031-15979-4_19.

[FS87] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *CRYPTO'86*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer, Heidelberg, Aug. 1987, pp. 186–194. DOI: 10.1007/3-540-47721-7_12.

[FGS07] Matthieu Finiasz, Philippe Gaborit, and Nicolas Sendrier. "Improved Fast Syndrome Based Cryptographic Hash Functions". In: *ECRYPT Hash Workshop 2007*. 2007. URL: https://www.finiasz.net/research/2007/finiasz-gaborit-sendrier-ecrypt-hash-workshop07.pdf.

[FS09] Matthieu Finiasz and Nicolas Sendrier. "Security Bounds for the Design of Code-Based Cryptosystems". In: *ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. LNCS. Springer, Heidelberg, Dec. 2009, pp. 88–105. DOI: 10.1007/978-3-642-10366-7_6.

[FGP+18] Pierre-Alain Fouque, Pierrick Gaudry, Thomas Pornin, and Thomas Prest. "Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU". In: *Submission to the NIST Post-Quantum Cryptography Standardization Project*. Available online: https://falcon-sign.info/. 2018. URL: https://falcon-sign.info/.

[GSR23] Philippe Gaborit, Julien Schrek, and Olivier Ruatta. *PERK: A Code-Based Post-Quantum Digital Signature Scheme*. Tech. rep. Additional Round 1 Submission. National Institute of Standards and Technology, 2023. URL: https://pqc-perk.org/.

[Gal68] Robert G Gallager. *Information theory and reliable communication*. Vol. 588. Springer, 1968.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. "Attribute-Based Encryption for Circuits from Multilinear Maps". In: *CRYPTO 2013, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. LNCS. Springer, Heidelberg, Aug. 2013, pp. 479–499. DOI: 10.1007/978-3-642-40084-1_27.

[GJK+07] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems". In: *Journal of Cryptology* 20.1 (Jan. 2007), pp. 51–83. DOI: 10.1007/s00145-006-0347-3.

[GI14] Niv Gilboa and Yuval Ishai. "Distributed Point Functions and Their Applications". In: *EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer, Heidelberg, May 2014, pp. 640–658. DOI: 10.1007/978-3-642-55220-5_35.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to Construct Random Functions". In: *Journal of the ACM* 33.4 (Oct. 1986), pp. 792–807.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof-Systems". In: *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*. 1985, pp. 291–304. DOI: 10.1145/22145.22178.

[Gro96] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*. Philadelphia, PA, USA: ACM, 1996, pp. 212–219. DOI: 10.1145/237814.237866. URL: https://doi.org/10.1145/237814.237866.

[GPS21] Shay Gueron, Edoardo Persichetti, and Paolo Santini. *Designing a Practical Code-based Signature Scheme from Zero-Knowledge Proofs with Trusted Setup*. Cryptology ePrint Archive, Report 2021/1020. https://eprint.iacr.org/2021/1020. 2021.

[GKW+20] Chun Guo, Jonathan Katz, Xiao Wang, and Yu Yu. "Efficient and Secure Multiparty Computation from Fixed-Key Block Ciphers". In: *2020 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2020, pp. 825–841. DOI: 10.1109/SP40000.2020.00016.

[GYW+23] Xiaojie Guo, Kang Yang, Xiao Wang, Wenhao Zhang, Xiang Xie, Jiang Zhang, and Zheli Liu. "Half-Tree: Halving the Cost of Tree Expansion in COT and DPF". In: *EUROCRYPT 2023, Part I*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14004. LNCS. Springer, Heidelberg, Apr. 2023, pp. 330–362. DOI: 10.1007/978-3-031-30545-0_12.

[GKS24] Kamil Doruk Gur, Jonathan Katz, and Tjerand Silde. "Two-Round Threshold Lattice-Based Signatures from Threshold Homomorphic Encryption". In: *Post-Quantum Cryptography: 15th International Workshop, PQCrypto 2024, Proceedings, Part II*. Oxford, UK: Springer-Verlag, June 2024, pp. 266–300.

[HOS+18] Carmit Hazay, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. "TinyKeys: A New Approach to Efficient Multi-Party Computation". In: *CRYPTO 2018, Part III*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10993. LNCS. Springer, Heidelberg, Aug. 2018, pp. 3–33. DOI: 10.1007/978-3-319-96878-0_1.

[HJ24] Janik Huth and Antoine Joux. *VOLE-in-the-head signatures from Subfield Bilinear Collisions*. Cryptology ePrint Archive, Paper 2024/1537. 2024. URL: https://eprint.iacr.org/2024/1537.

[IKO+07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. "Zero-knowledge from secure multiparty computation". In: *39th ACM STOC*. Ed. by David S. Johnson and Uriel Feige. ACM Press, June 2007, pp. 21–30. DOI: 10.1145/1250790.1250794.

[JL00] Stanislaw Jarecki and Anna Lysyanskaya. "Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures". In: *EUROCRYPT 2000*. Ed. by Bart Preneel. Vol. 1807. LNCS. Springer, Heidelberg, May 2000, pp. 221–242. DOI: 10.1007/3-540-45539-6_16.

[KT17] Ghazal Kachigar and Jean-Pierre Tillich. "Quantum Information Set Decoding Algorithms". In: *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017*. Ed. by Tanja Lange and Tsuyoshi Takagi. Springer, Heidelberg, 2017, pp. 69–89. DOI: 10.1007/978-3-319-59879-6_5.

[KZ20a] Daniel Kales and Greg Zaverucha. "An Attack on Some Signature Schemes Constructed from Five-Pass Identification Schemes". In: *CANS 20*. Ed. by Stephan Krenn, Haya Shulman, and Serge Vaudenay. Vol. 12579. LNCS. Springer, Heidelberg, Dec. 2020, pp. 3–22. DOI: 10.1007/978-3-030-65411-5_1.

[KZ20b] Daniel Kales and Greg Zaverucha. "Improving the Performance of the Picnic Signature Scheme". In: *IACR TCHES* 2020.4 (2020). https://tches.iacr.org/index.php/TCHES/article/view/8680, pp. 154–188. ISSN: 2569-2925. DOI: 10.13154/tches.v2020.i4.154-188.

[KRT24] Shuichi Katsumata, Michael Reichle, and Kaoru Takemure. "Adaptively Secure 5 Round Threshold Signatures from MLWE/MSIS and DL with Rewinding". In: *Annual International Cryptology Conference (CRYPTO 2024)*. Springer, 2024, pp. 459–491.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. "Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures". In: *ACM CCS 2018*. Ed. by David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang. ACM Press, Oct. 2018, pp. 525–537. DOI: 10.1145/3243734.3243805.

[KSW+97] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. "Secure Applications of Low-Entropy Keys". In: *Proceedings of the International Workshop on Information Security (ISW)*. Vol. 1396. Lecture Notes in Computer Science. Springer, 1997, pp. 121–134. DOI: 10.1007/BFb0052230. URL: https://link.springer.com/chapter/10.1007/BFb0052230.

[KPT+13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. "Delegatable pseudorandom functions and applications". In: *ACM CCS 2013*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. ACM Press, Nov. 2013, pp. 669–684. DOI: 10.1145/2508859.2516668.

[Kir11] Paul Kirchner. *Improved Generalized Birthday Attack*. Cryptology ePrint Archive, Report 2011/377. https://eprint.iacr.org/2011/377. 2011.

[KBC97] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. "HMAC: Keyed-Hashing for Message Authentication". In: *RFC* 2104 (1997), pp. 1–11. URL: https://api.semanticscholar.org/CorpusID:33812837.

[LB88] Pil Joong Lee and Ernest F Brickell. "An observation on the security of McEliece's public-key cryptosystem". In: *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer. 1988, pp. 275–280.

[Leo88] Jeffrey S Leon. "A probabilistic algorithm for computing minimum weights of large error-correcting codes". In: *IEEE Transactions on Information Theory* 34.5 (1988), pp. 1354–1359.

[LMØ+23] Fukang Liu, Mohammad Mahzoun, Morten Øygarden, and Willi Meier. *FAEST: Fault Aware Embedded Signature Technology*. Tech. rep. Additional Round 1 Submission. National Institute of Standards and Technology, 2023. URL: https://faest.info/.

[LR88] Michael Luby and Charles Rackoff. "How to Construct Pseudorandom Permutations from Pseudorandom Functions". In: *SIAM Journal on Computing* 17.2 (1988), pp. 373–386.

[Mat94] Mitsuru Matsui. "Linear Cryptanalysis Method for DES Cipher". In: *EUROCRYPT'93*. Ed. by Tor Helleseth. Vol. 765. LNCS. Springer, Heidelberg, May 1994, pp. 386–397. DOI: 10.1007/3-540-48285-7_33.

[MMT11] Alexander May, Alexander Meurer, and Enrico Thomae. "Decoding Random Linear Codes in $\tilde{\mathcal{O}}(2^{0.054n})$". In: *ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. LNCS. Springer, Heidelberg, Dec. 2011, pp. 107–124. DOI: 10.1007/978-3-642-25385-0_6.

[MO15] Alexander May and Ilya Ozerov. "On Computing Nearest Neighbors with Applications to Decoding of Binary Linear Codes". In: *EUROCRYPT 2015, Part I*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, Heidelberg, Apr. 2015, pp. 203–228. DOI: 10.1007/978-3-662-46800-5_9.

[McE78] Robert J. McEliece. *A public-key cryptosystem based on algebraic coding theory*. The Deep Space Network Progress Report 42-44. https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF. Jet Propulsion Laboratory, California Institute of Technology, Jan. 1978, pp. 114–116.

[Mer89] Ralph C. Merkle. "A Certified Digital Signature". In: *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*. Vol. 435. Lecture Notes in Computer Science. Springer, 1989, pp. 218–238. DOI: 10.1007/0-387-34805-0_21.

[MDC+11] Mohammed Meziani, Özgür Dagdelen, Pierre-Louis Cayrel, and Sidi Mohamed El Yousfi Alaoui. "S-FSB: An Improved Variant of the FSB Hash Family". In: *Information Security and Assurance (ISA 2011)*. Vol. 200. Communications in Computer and Information Science. Springer, 2011, pp. 132–145. DOI: 10.1007/978-3-642-23141-4_13. URL: https://link.springer.com/chapter/10.1007/978-3-642-23141-4_13.

[MS09] Lorenz Minder and Alistair Sinclair. "The extended k-tree algorithm". In: *20th SODA*. Ed. by Claire Mathieu. ACM-SIAM, Jan. 2009, pp. 586–595.

[MSY21] Jean-Pierre Münch, Thomas Schneider, and Hossein Yalame. *VASA: Vector AES Instructions for Security Applications*. Cryptology ePrint Archive, Report 2021/1493. https://eprint.iacr.org/2021/1493. 2021.

[Nak08] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: *Bitcoin.org* (2008). URL: https://bitcoin.org/bitcoin.pdf.

[Nat99] National Institute of Standards and Technology. *Data Encryption Standard (DES)*. Federal Information Processing Standards Publication FIPS PUB 46-3. U.S. Department of Commerce, 1999. URL: https://csrc.nist.gov/pubs/fips/46-3/final.

[Nat15a] National Institute of Standards and Technology. *Secure Hash Standard (SHS)*. Federal Information Processing Standards Publication FIPS PUB 180-4. U.S. Department of Commerce, 2015. URL: https://doi.org/10.6028/NIST.FIPS.180-4.

[Nat15b] National Institute of Standards and Technology. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. Federal Information Processing Standards Publication FIPS PUB 202. Defines SHA-3, including SHA3-224, SHA3-256, SHA3-384, SHA3-512, and extendable-output functions (SHAKE). U.S. Department of Commerce, 2015. URL: https://doi.org/10.6028/NIST.FIPS.202.

[Natng] National Institute of Standards and Technology. *Post-Quantum Cryptography Standardization Project*. Project to standardize quantum-resistant cryptographic algorithms. 2016–ongoing. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography.

[NCB11] Robert Niebuhr, Pierre-Louis Cayrel, and Johannes Buchmann. "Improving the efficiency of Generalized Birthday Attacks against certain structured cryptosystems". In: *WCC 2011-Workshop on coding and cryptography*. 2011, pp. 163–172.

[OTX24] Ying Ouyang, Deng Tang, and Yanhong Xu. "Code-Based Zero-Knowledge from VOLE-in-the-Head and Their Applications: Simpler, Faster, and Smaller". In: *ASIACRYPT 2024 (to appear)*. 2024. URL: https://eprint.iacr.org/2024/1414.

[Pat09] Jacques Patarin. "The "Coefficients H" Technique". In: *Selected Areas in Cryptography – SAC 2009*. Vol. 5867. Lecture Notes in Computer Science. Springer, 2009, pp. 328–345. DOI: 10.1007/978-3-642-05445-7_21.

[Pei10] Chris Peikert. "An Efficient and Parallel Gaussian Sampler for Lattices". In: *CRYPTO 2010*. Ed. by Tal Rabin. Vol. 6223. LNCS. Springer, Heidelberg, Aug. 2010, pp. 80–97. DOI: 10.1007/978-3-642-14623-7_5.

[PL22] Ba-Duc Pham and Pierre Loidreau. "An analysis of Coggia-Couvreur Attack on Loidreau's Rank-metric public-key encryption scheme in the general case". In: *WCC 2022 : Twelth International Workshop on Coding and Cryptography*. 2022. URL: https://www.wcc2022.uni-rostock.de/storages/uni-rostock/Tagungen/WCC2022/Papers/WCC_2022_paper_38.pdf.

[PKM+24] Rafael del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani Saarinen. *Threshold Raccoon: Practical Threshold Signatures from Standard Lattice Assumptions*. Cryptology ePrint Archive, Paper 2024/184. 2024. URL: https://eprint.iacr.org/2024/184.

[Pra62] Eugene Prange. “The use of information sets in decoding cylcic codes”. In: *IRE Transactpons on Information Theory* 8.5 (1962), pp. 5–9.

[Red27] J Howard Redfield. “The theory of group-reduced distributions”. In: *American Journal of Mathematics* 49.3 (1927), pp. 433–455.

[Reg06] Oded Regev. “Lattice-Based Cryptography (Invited Talk)”. In: *CRYPTO 2006*. Ed. by Cynthia Dwork. Vol. 4117. LNCS. Springer, Heidelberg, Aug. 2006, pp. 131–141. DOI: 10.1007/11818175_8.

[RS21] Peter Rindal and Phillipp Schoppmann. “VOLE-PSI: Fast OPRF and Circuit-PSI from Vector-OLE”. In: *EUROCRYPT 2021, Part II*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12697. LNCS. Springer, Heidelberg, Oct. 2021, pp. 901–930. DOI: 10.1007/978-3-030-77886-6_31.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. In: *Communications of the ACM* 21.2 (1978), pp. 120–126. DOI: 10.1145/359340.359342.

[RW19] Dragos Rotaru and Tim Wood. *MArBled Circuits: Mixing Arithmetic and Boolean Circuits with Active Security*. Cryptology ePrint Archive, Report 2019/207. https://eprint.iacr.org/2019/207. 2019.

[Roy22] Lawrence Roy. “SoftSpokenOT: Quieter OT Extension from Small-Field Silent VOLE in the Minicrypt Model”. In: *CRYPTO 2022, Part I*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13507. LNCS. Springer, Heidelberg, Aug. 2022, pp. 657–687. DOI: 10.1007/978-3-031-15802-5_23.

[Saa07] Markku-Juhani Olavi Saarinen. “Linearization Attacks Against Syndrome Based Hashes”. In: *INDOCRYPT 2007*. Ed. by K. Srinathan, C. Pandu Rangan, and Moti Yung. Vol. 4859. LNCS. Springer, Heidelberg, Dec. 2007, pp. 1–9.

[Sho97] Peter W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 1095-7111. DOI: 10.1137/s0097539795293172. URL: http://dx.doi.org/10.1137/S0097539795293172.

[Sho00] Victor Shoup. “Practical Threshold Signatures”. In: *Advances in Cryptology – EUROCRYPT 2000*. Vol. 1807. Lecture Notes in Computer Science. Springer, 2000, pp. 207–220. DOI: 10.1007/3-540-45539-6_15. URL: https://link.springer.com/chapter/10.1007/3-540-45539-6_15.

[Ste88] Jacques Stern. “A method for finding codewords of small weight”. In: *International Colloquium on Coding Theory and Applications*. Springer. 1988, pp. 106–113.

[Ste94a] Jacques Stern. "A New Identification Scheme Based on Syndrome Decoding". In: *CRYPTO'93*. Ed. by Douglas R. Stinson. Vol. 773. LNCS. Springer, Heidelberg, Aug. 1994, pp. 13–21. DOI: 10.1007/3-540-48329-2_2.

[Ste94b] Jacques Stern. "Designing Identification Schemes with Keys of Short Size". In: *CRYPTO'94*. Ed. by Yvo Desmedt. Vol. 839. LNCS. Springer, Heidelberg, Aug. 1994, pp. 164–173. DOI: 10.1007/3-540-48658-5_18.

[SS01] Douglas R. Stinson and Reto Strobl. "Provably Secure Distributed Schnorr Signatures and a (t, n) Threshold Scheme for Implicit Certificates". In: *Information Security and Privacy*. Ed. by Vijay Varadharajan and Yi Mu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 417–434. ISBN: 978-3-540-47719-8.

[TSZ19] Teik Guan Tan, Pawel Szalachowski, and Jianying Zhou. *Challenges of Post-Quantum Digital Signing in Real-world Applications: A Survey*. Cryptology ePrint Archive, Paper 2019/1374. 2019. URL: https://eprint.iacr.org/2019/1374.

[WRK17a] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. "Authenticated Garbling and Efficient Maliciously Secure Two-Party Computation". In: *ACM CCS 2017*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM Press, Oct. 2017, pp. 21–37. DOI: 10.1145/3133956.3134053.

[WRK17b] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. "Global-Scale Secure Multiparty Computation". In: *ACM CCS 2017*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM Press, Oct. 2017, pp. 39–56. DOI: 10.1145/3133956.3133979.

[WYK+21] Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. "Wolverine: Fast, Scalable, and Communication-Efficient Zero-Knowledge Proofs for Boolean and Arithmetic Circuits". In: *2021 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2021, pp. 1074–1091. DOI: 10.1109/SP40001.2021.00056.

[Woo14] Gavin Wood. *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. Tech. rep. Ethereum Project, 2014. URL: https://ethereum.github.io/yellowpaper/paper.pdf.

[YSW+21] Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. "QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field". In: *ACM CCS 2021*. Ed. by Giovanni Vigna and Elaine Shi. ACM Press, Nov. 2021, pp. 2986–3001. DOI: 10.1145/3460120.3484556.

[YWL+20] Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. "Ferret: Fast Extension for Correlated OT with Small Communication". In: *ACM CCS 2020*. Ed. by Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna. ACM Press, Nov. 2020, pp. 1607–1626. DOI: 10.1145/3372297.3417276.

[Yao82] Andrew C. Yao. "Theory and application of trapdoor functions". In: *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*. 1982, pp. 80–91. DOI: 10.1109/SFCS.1982.45.

[Yao08] Andrew Chi-Chih Yao. "Some Perspectives on Complexity-Based Cryptography (Invited Talk)". In: *ASIACRYPT 2008*. Ed. by Josef Pieprzyk. Vol. 5350. LNCS. Springer, Heidelberg, Dec. 2008, p. 54. DOI: 10.1007/978-3-540-89255-7_4.

[ZCD+20] Greg Zaverucha et al. *Picnic*. Tech. rep. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. National Institute of Standards and Technology, 2020.

[ZLH24] Yunxiao Zhou, Shengli Liu, and Shuai Han. "Multi-hop Fine-Grained Proxy Re-encryption". In: *Public-Key Cryptography – PKC 2024*. Ed. by Qiang Tang and Vanessa Teague. Cham: Springer Nature Switzerland, 2024, pp. 161–192. ISBN: 978-3-031-57728-4.