DISSERTATIONES
INFORMATICAE
UNIVERSITATIS
TARTUENSIS
18

JANNO SIIM

Non-Interactive Shuffle Arguments

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

18

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

18

JANNO SIIM

Non-Interactive Shuffle Arguments

UNIVERSITY OF TARTU
Press

Institute of Computer Science, Faculty of Science and Technology, University of Tartu, Estonia.

Dissertation has been accepted for the commencement of the degree of Doctor of Philosophy (PhD) in Computer Science on the 16th of June, 2020 by the Council of the Institute of Computer Science, University of Tartu.

Supervisor

Helger Lipmaa	
Simula UiB	University of Tartu
Bergen, Norway	Tartu, Estonia

Opponents

Eran Tromer	
Tel Aviv University	Columbia University
Tel Aviv, Israel	New York, USA

Dario Fiore
IMDEA Software Institute
Madrid, Spain

The public defense will take place on August 28, 2020 at 16:15 in Zoom.

The publication of this dissertation was financed by the Institute of Computer Science, University of Tartu.

ISSN 2613-5906
ISBN 978-9949-03-377-5 (print)
ISBN 978-9949-03-378-2 (PDF)

University of Tartu Press
`http://www.tyk.ee/`

ABSTRACT

A fundamental requirement for all democratic governments is a secure voting system. In recent years some countries, like Estonia and Switzerland, have adopted internet voting (i-voting) and many more have experimented (e.g., Norway and Australia) or have plans to adopt it in the future (e.g., Lithuania and Russia). I-voting has the potential to offer better convenience, lower administrative costs, and higher voter turnout, but this comes with increased security concerns and significant technical challenges.

Consider the simple procedure of shaking the ballot box to mix the order of the ballots. It is far from obvious how to achieve an equivalent result with encrypted digital ballots. Who should perform the mixing procedure? How to guarantee that it was performed correctly? Is it possible that no one can trace the ballots?

This problem can be solved with a distributed system called a mix-network. The idea is to let each peer in the mix-network shuffle (permute and rerandomize) the ciphertexts. This makes it computationally hard to trace the input ciphertexts to the output ciphertexts given that at least one peer is honest. However, each peer should also give a proof that the shuffling was done correctly to avoid substitution attacks. The proof has to be hard to forge *(sound)* and should leak nothing but the truth of the statement *(zero-knowledge)*. Such proofs are called zero-knowledge shuffle arguments, and in this thesis, we study their constructions. Importantly, we avoid the heuristic security model, used in many of the previous works, which (incorrectly) treats a cryptographic hash function as a truly random function. We show that it possible to construct shuffle arguments, that are efficient enough for large-scale elections, in other security models than the random oracle model.

First, we construct a very efficient non-interactive shuffle argument that avoids the random oracle model and instead uses the generic group model. This is achieved by combining several recent tools like quasi-adaptive zero-knowledge arguments and SNARKs. We implement it and observe practical efficiency for large-scale elections: the proving time for 100,000 ciphertexts is less than a minute, and verification time is less than 1.5 minutes on modest hardware. Unfortunately, security requires that the prover and the verifier have access to a trusted common reference string (CRS).

Secondly, we study how to reduce trust assumptions. There are efficient multi-party computation (MPC) protocols for generating CRSs for a large class of arguments, but they require the random oracle model. We improve upon one such protocol and, among other results, remove the requirement for the random oracle. We prove the security of this protocol in the universal composability setting.

Thirdly, we modify our shuffle argument to be applicable to the above MPC protocol. This guarantees both soundness and zero-knowledge as long as at least one peer in the MPC protocol is honest. We go one step further and show how to get zero knowledge even if all the peers are malicious. Additionally, we simplify the argument construction and prove its security based on weaker assumptions.

CONTENTS

LIST OF ABBREVIATIONS

AGM : Algebraic Group Model
BDH-KE : Bilinear Diffie-Hellman Knowledge of Exponent
CDH Computational Diffie-Hellman
CRS : Common Reference String
DDH : Decisional Diffie-Hellman
DL : Discrete Logarithm
GGM : Generic Group Model
IND-CPA : INDistinguishability under Chosen Plain Attack
i-voting : internet voting
KerMDH : Kernel Matrix Diffie-Hellman
NIZK : Non-Interactive Zero-Knowledge
MP : Multi-Pedersen
PDL : Power Discrete Logarithm
PPT : Probabilistic Polynomial Time
pv-RRCCA publicly verifiable Rerandomizable RCCA
RCCA : Replayable Chosen Ciphertext Attack
RO : Random Oracle
SNARK : Succinct Non-interactive ARgument of Knowledge
SPDL : Symmetric Power Discrete Logarithm
SXDH : Symmetric External Diffie-Hellman
UC : Universal Composability
WI : Witness Indistinguishability
ZK : Zero-Knowledge
Sub-ZK : Subversion ZK

LIST OF ORIGINAL PUBLICATIONS

Publications included in the thesis

1. Prastudy Fauzi, Helger Lipmaa, Janno Siim, and Michał Zając. An Efficient Pairing-based Shuffle Argument. In Tsuyoshi Takagi and Thomas Peyrin, editors, ASIACRYPT 2017, Part II, volume 10625 of LNCS, pages 97–127. Springer, Heidelberg, December 2017.
2. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, Janno Siim, and Michał Zając. UC-secure CRS generation for SNARKs. In AFRICACRYPT 19, LNCS, pages 99–117. Springer, Heidelberg, 2019.
3. Antonis Aggelakis, Prastudy Fauzi, Georgios Korfatis, Panos Louridas, Foteinos Mergoupis-Anagnou, Janno Siim, and Michał Zając. A Non-interactive Shuffle Argument With Low Trust Assumptions. In CT-RSA 2020, LNCS, volume 12006, pages 667-692. Springer, Cham, 2020.

Publications not included in the thesis

1. Aggelos Kiayias, Annabell Kuldmaa, Helger Lipmaa, Janno Siim, and Thomas Zacharias. On the Security Properties of E-voting Bulletin Boards. In Dario Catalano and Roberto De Prisco, editors, SCN 18, volume 11035 of LNCS, pages 505–523. Springer, Heidelberg, September 2018.
2. Sven Heiberg, Janno Siim, Ivo Kubjas, Jan Willemson. On Trade-offs of Applying Block Chains for Electronic Voting Bulletin Boards. Proceedings of the Third International Joint Conference on Electronic Voting E-Vote-ID 2018: E-Vote-ID 2018, October 2-5, 2018, Bregenz, Austria. Ed. Robert Krimmer, Melanie Volkamer, Véronique Cortier, David Duenas-Cid, Rajeev Goré, Manik Hapsara, Reto Koenig, Steven Martin, Ronan McDermott, Peter Roenne, Uwe Serdült, Tomasz Truderung. TUT Press, 259-276.
3. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, Janno Siim, and Michał Zając. DL-extractable UC-commitment Schemes. In ACNS 19, LNCS, pages 385–405. Springer, Heidelberg, 2019.
4. Behzad Abdolmaleki, Helger Lipmaa, Janno Siim, and Michal Zajac. On QA-NIZK in the BPK model. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, PKC 2020, Part I, volume 12110 of LNCS, pages 590–620. Springer, Heidelberg, May 2020.

1. INTRODUCTION

In recent decades, there has been a large shift towards moving all kinds of services online. One exception to this has been voting which has shown strong resistance to digitalization, mainly since it opens up new ways to attack elections. For example, Norway, Australia, France, and Canada have all seen setbacks after the initial trials of internet voting (i-voting). Still, some countries have adopted i-voting. Most successful has been Estonia that allows both municipal and parliamentary elections online, and Switzerland that also has i-voting in several cantons.

Like traditional voting, a secure i-voting system should guarantee anonymity for voters. One way to achieve this is encryption, but it alone is not enough since ballots usually also include some authenticating information, for example, a digital signature. Paper-based voting solves this problem by having voters place ballots into a ballot box (after the voter has been identified) which is then shaken to mix the order of ballots. In i-voting, we should send ciphertexts through an anonymous communication channel before decrypting. Applications for anonymous channels go even far beyond just i-voting and include for example anonymouse e-mail [Kat], private information retrieval [IKOS06], and anonymous cryptocurrencies [BNM+14, KRDO17].

Chaum [Cha81] proposed a mix-network protocol for implementing an anonymous channel. Mix-network is a distributed system consisting of N peers. Peers take turns to shuffle the ciphertexts which involves randomizing the appearance of each ciphertext (while keeping the same message) and then randomly permuting their order. Let us assume that at least one peer is honest and the rest are semi-honest, that is, they execute the protocol correctly but study and share all the leaked information. In such a case, it will be computationally hard to match the initial ciphertexts to the final ciphertexts output by the N-th peer.

However, the semi-honest model is not sufficient for applications like i-voting considering that any peer could easily substitute some of the ciphertexts while remaining undetected. We can tolerate $N-1$ fully malicious peers if we additionally require each peer to provide proof of correct shuffle execution. Of course, the proof should not leak any information about the permutation or randomness used for blinding the ciphertexts. Otherwise, we would undermine the original goal of the mix-network. This task is well suited for a cryptographic tool called a zero-knowledge argument (or proof)[1] [GMR89]. Zero-knowledge arguments allow proving the validity of a statement while leaking no other information. Arguments can either need interaction between the prover and the verifier or be non-interactive, i.e., prover just outputs a single message which can then be verified by anyone. In the i-voting scenario, we would typically prefer non-interactive shuffle arguments since this allows election auditing even if peers (together with their

[1]Proof systems are unforgeable even against computationally unbounded adversaries whereas argument systems tolerate only efficient adversaries. In this thesis we only consider arguments.

private information) are not available anymore.

There has been extensive research into interactive zero-knowledge shuffle arguments [SK95,Abe99,Nef01,FS01,FMM$^+$03,GI08,TW10,Gro10,BG12]. More recent of these constructions [Gro10, BG12] rely on standard assumptions (e.g., discrete logarithm assumption) and achieve the efficiency that could be practical for large-scale elections. Moreover, most of them are public coin protocols, meaning that verifier's messages are chosen uniformly randomly and independently from prover's messages. This makes it possible to apply the Fiat-Shamir tranform [FS87] which makes those arguments non-interactive. Unfortunately, the resulting argument is secure only in the random oracle (RO) model which essentially assumes a black-box implementation of a uniformly random function. In practice, the uniformly random function is substituted with a secure hash function. However, at least on a theoretical level, it is known that this substitution is unwarranted. There are contrived protocols that are secure in the RO model, but trivially insecure when instantiated with a secure hash function [Bar01, GK03, BDG$^+$13, BBH$^+$19]. Similar attacks are not known for protocols that have been designed with "honest indentions", but a distinction between a contrived protocol and a non-contrived protocol is, of course, subjective and thus it is better not to rely on such assumptions whenever possible.

More recently, several non-interactive shuffle arguments have been proposed which do not rely on the RO model [GL07, LZ13, GR16, FL16, FLZ16, FFHR19]. For ease, let us call such protocols RO-free. All of those arguments (actually, even many of the interactive arguments) require a common reference string (CRS) model [BFM88], i.e., it is assumed that a trusted party picks a bitstring crs from some distribution $\mathscr{D}$ and provides it to the prover and the verifier. Due to this reason, it is unclear how to apply those arguments in practice where we usually cannot trust any parties. Moreover, the efficiency of the current RO-free arguments is noticeably worse than the efficiency of the best RO model arguments [Gro10, BG12]. Efficiency and trust requirements of RO-free shuffle arguments are the main issues that we tackle in this thesis.

1.1. Contributions and Outline

Thesis follows three published papers respectively summarized in Chapter 3, Chapter 4, and Chapter 5. Background for these result is given in Chapter 2 and final conclusions and open questions are discussed in Chapter 6.

Since the thesis is based on multi-author papers, then we separately mention some contributions from the author of this thesis. Of course, it should be understood that the distinction of who did what is not so clear-cut in collaborative research. It is rare that one author achieves a result of any significance without being affected by one's co-authors at all. Thus, this distinction should be viewed such that the noted results are the ones where the author had the biggest impact among all of the co-authors.

Chapter 3: An Efficient Shuffle Argument. We summarize the work of Fauzi, Lipmaa, Siim, and Zając [FLSZ17a]. More precisely, we follow the full version [FLSZ17b] since the conference version contained a subtle flaw related to zero-knowledge property. The result builds upon the previous work of Fauzi, Lipmaa, Zając [FLZ16] and proposes so far the most efficient non-interactive zero-knowledge shuffle argument in the CRS model. We denote it by $\mathcal{S}$ in the following. Our argument uses the standard ElGamal cryptosystem compared to more specialized cryptosystems used in some of the other constructions [GL07, FLZ16,FFHR19]. On a technical level, we mix several previously known but very different arguments like zero-knowledge Succinct Non-interactive ARguments of Knowledge (zk-SNARKs) from Danezis et al. [DFGK14], quasi-adaptive zero-knowledge argument by Kiltz and Wee [KW15], and then some more shuffle specific ideas from Groth and Lu [GL07] and from other previous shuffle arguments [LZ13, FL16, FLZ16]. The final argument has perfect zero-knowledge and standard notion of computational soundness. However, some of the sub-arguments are proved in the generic group model (GGM) which allows an adversary to only make group operations while the group structure is hidden.

We optimize each sub-argument as much as possible and also apply batching techniques [BGR98] which amortize the verifying cost for multiple similar sub-arguments. We provide a well-optimized C++ implementation and see that for 100,000 ciphertexts the proving time is under 1 minute and verification time under 1.5 minutes on a laptop computer. This is close to the running time reported by Bayer and Groth [BG12] (2 minutes for proving and verifying on similar hardware) for one of the most efficient interactive shuffle arguments.

Author's contributions include optimizing sub-arguments, applying batching techniques, and implementing the argument, but also, for example, reproving zero-knowledge property in the final version of the paper after a subtle flaw was found.

Chapter 4: UC-Secure CRS Generation For NIZK. This chapter is based on the paper by Abdolmaleki, Baghery, Lipmaa, Siim and Zając [ABL$^+$19b]. The shuffle argument $\mathcal{S}$ is secure only in the CRS model. Motivated by this, we are interested in reducing the trust requirements of non-interactive zero-knowledge (NIZK) arguments in the CRS model. Common folklore's suggestion is to generate the CRS with some secure multi-party protocol to distribute trust among multiple peers. Many of the efficient non-interactive zero-knowledge (NIZK) arguments, including our shuffle argument, operate in the pairing-based setting [2] and require a relatively large CRS. Standard secure multi-party protocols are not efficient in this case if we also want to have a large number of peers in a fully malicious setting. Ben-Sasson et al. [BCG$^+$14] designed a special-purpose multi-party protocol for CRS generation. Their protocol works for a large class of pairing-

[2]Pairings are bilinear maps $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that allow computing $\hat{e}(g_1^x, g_2^y) = g_T^{xy}$ where $\mathbb{G}_i$ is some group with a generator g_i for $i \in \{1, 2, T\}$. We cover them in more detail in Chapter 2.

based NIZK arguments, is much more efficient for CRS generation compared to standard multi-party protocols, and requires only 1 peer to be honest. Unfortunately, it also assumes the RO model and is thus unsuitable for the argument $\mathcal{S}$.

In this work, we modified Ben-Sasson et al.'s protocol and make it RO-free. A key tool that we use is a discrete logarithm (DL) extractable commitment by Abdolmaleki et al. [ABL$^+$19a] which opens to g^x but allows extraction of x in the security proof. Essentially, in the first phase of Ben-Sasson et al.'s protocol, each peer has to reveal some shares of the form g^x. However, peers must pick those shares independently. The original protocol achieves this by forcing peers to output an RO-based NIZK argument which proves that they know x. We instead run a coin-tossing protocol where all peers first commit to their share x with a DL-extractable commitment in the first round and then open it to g^x in the next round. Since the DL-extractable commitment we use is universally composable (UC) [Can01], then it is also non-malleable and we get that shares are chosen independently. Extractability allows us to simulate the protocol transcript and prove that the whole protocol is also UC-secure. We note that the DL-extractable commitment itself might also need a trust assumption but this depends on the concrete instantiation and is hopefully a weaker assumption than a trusted CRS.

THe UC-security guarantees that our CRS generation protocol can be safely composed with the NIZK argument without affecting the security properties of the NIZK itself. Moreover, we introduce several efficiency improvements and show that class of arguments $\mathscr{C}$ that we can cover includes the most efficient SNARK by Groth [Gro16].

The author's contributions here include (among others) proposing the ideal functionality for the CRS generation protocol and proving the security of the protocol.

Chapter 5: Transparent Shuffle Argument. The final result is based on the work of Aggelakis, Fauzi, Korfatis, Louridas, Mergoupis-Anagnou, Siim, and Zając [AFK$^+$20]. Our main objective with the CRS generation protocol was to distribute the trust of argument $\mathcal{S}$. It turns out that the argument $\mathcal{S}$ is outside of the class $\mathscr{C}$ and thus we cannot directly apply the CRS generation protocol. We have to modify the argument $\mathcal{S}$ and construct a new argument $\mathcal{S}^+$ that we call a transparent shuffle argument due significantly reduced trust assumptions. Mainly we modify the structure of the CRS, but we also simplify the way that different sub-arguments are combined. After making several non-trivial changes to the argument, we also need to reprove the soundness and zero-knowledge properties. Here, we also manage to make some improvements. We prove the soundness of some of the sub-arguments under weaker assumptions compared to Fauzi et al. [FLSZ17a] and other arguments that had proof in GGM we now prove in a weaker algebraic group model (AGM) [FKL18] .

CRS generation protocol guarantees soundness and zero-knowledge if at least one of the peers is honest. Turns out that it is possible to go one step further. Following the works of [BFS16, ABLZ17, Fuc18] we construct a CRS verification algorithm such that if it accepts the CRS, then the zero-knowledge property will hold. This is known as statistical subversion zero-knowledge in [BFS16, ABLZ17, Fuc18] or equivalently no-auxiliary-string non-black-box zero-knowledge in the bare public-key model as was recently defined in [ALSZ18]. Importantly, it means that zero-knowledge holds even if all the peers in the CRS generation protocol are malicious. Finally, we also implement a complete set of algorithms. That includes the prover's algorithm, verifier's algorithm, distributed CRS generation protocol, and the CRS verification algorithm.

The main contributions of the author were modifications of the CRS to make $\mathcal{S}^+$ suitable for the CRS generation protocol, coming up with the CRS verification algorithm, and proving many of the related results.

Claim of the Thesis. Based on the above results we formulate the main claim of this thesis:

> *It is possible to construct a non-interactive zero-knowledge shuffle argument that is* ***RO-free*** *and still* ***practical*** *for large-scale elections.*

The notion of practicality is slightly subjective, but we believe that it is well justified in our case. In particular, it follows from subsequent facts:

1. Shuffle arguments $\mathcal{S}$ and $\mathcal{S}^+$ are sufficiently efficient for large numbers of ciphertexts as was shown in [FLSZ17a].
2. Trust assumptions will not be a major obstacle in practice. As we showed in [AFK$^+$20], the argument $\mathcal{S}^+$ is secure if we only trust 1 out of N peers in the setup phase of the argument.
3. Argument $\mathcal{S}$ has been integrated to the Zeus i-voting system [TPLT13] which is extensively used in practice [BCD$^+$19]. Integration is in progress for the newer argument $\mathcal{S}^+$.

1.2. Related Work

We will briefly compare state-of-the-art art shuffle arguments to the results of this thesis. Efficiency wise the most interesting parameters are prover's and verifier's computational complexity, proof size, and CRS size. Since some shuffle arguments use specialized cryptosystems, occasionally with relatively long ciphertexts, then we will count shuffling complexity as part of the prover's overall complexity and ciphertext size as part of the proof size. This seems fair to us considering that a mixer will output both the proof and the shuffled ciphertexts.

We also consider several qualitative properties that include the type of cryptosystem, type of CRS, security assumptions, the flavor of soundness, and flavor of zero-knowledge. Since a mix-network is usually part of a larger system (say, part of an i-voting system), then it is preferable to have a relatively standard cryptosystem with short ciphertexts such as Elgamal. Uniformly random CRS is usually preferred over a more specialized distribution since it is easier to implement based on some natural source of randomness (sunspot data, stock market fluctuations, etc.). Perfect or statistical zero-knowledge is usually considered to be better than computational zero-knowledge since for the former we do not need to worry that some computational assumption gets broken.

There are numerous interactive shuffle arguments [SK95, Abe99, Nef01, FS01, FMM$^+$03, GI08, TW10, Gro10, BG12]. However, since our main interest is non-interactive RO-free shuffle arguments, then we will not focus on them too much. State-of-the-art interactive shuffles are [Gro10] by Groth which has linear complexity in all of the previously mentioned efficiency parameters (with quite small coefficients) and [BG12] by Bayer and Groth which has a sublinear proof size (excluding the ciphertext size). Both of them support for example Elgamal ciphertexts and require only discrete logarithm (DL) assumption[3].

The line of work on non-interactive shuffle arguments was started by Groth and Lu [GL07] that proposed the first construction secure without RO. They proved only culpable soundness[4] and used BBS cryptosystem [BBS04] which has ciphertext size of 3 group elements. As the main tool, they applied Groth-Sahai proofs [GS08] together with some novel assumptions. In their construction, all relevant efficiency parameters are linear in the number of ciphertexts n but mostly with relatively large constants.

Lipmaa and Zhang [LZ13] proposed two non-interactive shuffle arguments that achieve the usual notion of soundness, but with a (non-falsifiable, see Section 2.2) knowledge assumption. They relied on an even more complicated variant of BBS cryptosystem which has 6 group elements and is lifted (decrypting requires computing discrete logarithm). Efficiency-wise, one of the construction has CRS size $\Theta(\sqrt{n})$, but other parameters are superlinear and the second construction has linear parameters with smaller constants than [GL07] except for the CRS size.

Fauzi and Lipmaa [FL16] further improved efficiency compared to Lipmaa and Zhang's result. However, they use both knowledge assumptions and prove only culpable soundness. They give the first construction which works with standard non-lifted Elgamal cryptosystem. The unit vector argument used in our constructions is essentially borrowed from this work.

Fauzi, Lipmaa, and Zając [FLZ16] achieve significant improvement in verification time compared to previous works. However, they prove the whole argument in the generic group model (GGM) which restricts adversary's operations to group operations and hides the group structure. They use the non-standard lifted ILin cryptosystem [EHK$^+$13] which contains 3 group elements.

González and Ráfols [GR16] proposed an argument that uses standard Elgamal cryptosystem and soundness is based only on falsifiable assumptions which are considered more safe [Nao03, GK16]. It also achieves relatively good efficiency in all parameters except for CRS size which is $\Theta(n^2)$. Latter is, unfortunately, a big drawback and makes it impractical for large-scale i-voting systems.

[3]Elgamal cryptosystem itself, of course, requires decisional Diffie-Hellman (DDH) assumption.

[4]Culpable soundness restricts the adversary to have knowledge of his cheating. See Section 2.4 for further discussion and the formal definition.

Since the publication of our work [FLSZ17a] in 2017, Faonio et al. [FFHR19] proposed a mix-network which uses a specialized cryptosystem in a more fundamental way. First, they construct a re-randomizable RCCA (Replayable Chosen Ciphertext Attack)[5] [CKN03] secure cryptosystem. They make it publicly verifiable by including a Groth-Sahai proof and then use techniques similar to Faonio and Fiore [FF18] (which itself is RO-based mix-network) to construct a shuffle argument. Interestingly, CRS in their case is a relatively small uniformly random string which only depends on the number of mixers and their construction is UC-secure under DDH assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$ (SXDH). On a negative side, their publicly verifiable re-randomizable RCCA (pv-RRCCA) cryptosystem is relatively inefficient. For example the ciphertext size is 12 elements in $\mathbb{G}_1$, 11 elements in $\mathbb{G}_2$, and 4 elements in $\mathbb{G}_T$.

Finally, we mention a series of works [CKLM12, CKLM13b, CKLM13a] that construct controllably malleable shuffle arguments respectively from Groth-Sahai proofs [GS08], Groth and Lu shuffle, and SNARKs. Essentially the idea is to let each mixer mall the argument returned by the previous mixer and include a short proof of participation. The whole mix-network can be verified with a single argument that is returned by the last mixer. Importantly, the argument size grows only additively in the number mixers, i.e., the size is $\Theta(n+N)$ where N is the number of mixers and n is the number of input ciphertexts. All other arguments would have a collective argument size of $\Theta(nN)$ assuming that a single argument has size $\Theta(n)$. Their techniques seem to be fundamentally incompatible with Fiat-Shamir based arguments since hashing is non-malleable. However, it seems plausible that the constructions of this paper can also be extended to controlled malleability but with significantly better efficiency than the Groth and Lu argument. We leave this as an open problem.

Efficiency of the best non-interactive arguments is compared in Table 1 and other properties are compared in Table 2. We see that [Gro10, BG12] are still slightly more efficient than RO-free shuffles but the gap is small. Computational security assumptions are still less standard except for Faonio et al.'s construction which only requires SXDH assumption. Of course, we remind that [Gro10,BG12] also requires the RO model in the non-interactive setting which is only a heuristic. Arguably the best RO-free shuffle arguments are the constructions from this thesis and the construction of Faonio et al. Ours has better efficiency and uses a standard cryptosystem, but [FFHR19] has better security properties and a very short and uniform CRS.

[5] This is a slightly weaker version of CCA security which allows malleability if the message of the ciphertext stays the same.

Table 1. Efficiency comparison of state-of-the-art shuffles. Here n is the number of ciphertexts, N the number of mixers, E is the cost of exponentiation, and P is the cost of pairing. Prover's complexity includes shuffling time and proof size includes the size of the output ciphertexts. We drop constant terms.

	Prover efficiency	**Verifier efficiency**	**Proof size**	**CRS size**
[Gro10]	$8n \times E$	$6n \times E$	$2n \times \mathbb{G}$ $3n \times \mathbb{Z}_p$	$n \times \mathbb{G}$
[BG12]	$2n + 2n\log(\sqrt{n}) \times E$	$4n \times E$	$2n + 11\sqrt{n} \times \mathbb{G}$ $5\sqrt{n} \times \mathbb{Z}_p$	$\sqrt{n} \times \mathbb{G}$
[FL16]	$20n \times E$	$18n \times P$	$7n \times \mathbb{G}_1$ $2n \times \mathbb{G}_2$	$6n \times \mathbb{G}_1$ $2n \times \mathbb{G}_2$
[FLZ16]	$18n \times E$	$14n \times E$ $3n \times P$	$5n \times \mathbb{G}_1$ $4n \times \mathbb{G}_2$	$2n \times \mathbb{G}_1$ $n \times \mathbb{G}_2$
[GR16]	$13n \times E$	$13n \times P$	$4n \times \mathbb{G}_1$ $2n \times \mathbb{G}_2$	$n^2 + 24n \times \mathbb{G}_1$ $23n \times \mathbb{G}_2$
[FFHR19]	$72n \times E$ $5n \times P$	$22n \times P$	$12n \times \mathbb{G}_1$ $11n \times \mathbb{G}_2$ $4n \times \mathbb{G}_T$	$2N \times \mathbb{G}_1$ $2N \times \mathbb{G}_2$
$\mathcal{S}$ (Chapter 3)	$11n \times E$	$7n \times E$ $3n \times P$	$4n \times \mathbb{G}_1$ $3n \times \mathbb{G}_2$	$4n \times \mathbb{G}_1$ $n \times \mathbb{G}_2$
$\mathcal{S}^+$ (Chapter 5)	$11n \times E$	$7n \times E$ $3n \times P$	$4n \times \mathbb{G}_1$ $3n \times \mathbb{G}_2$	$5n \times \mathbb{G}_1$ $n \times \mathbb{G}_2$

Table 2. Qualitative properties of state-of-the-art non-interactive shuffle arguments

	Encryption	**Assumption**	**Soundness**	**ZK**	**CRS**
[Gro10]	Elgamal	ROM, DL	standard	statistical	uniform
[BG12]	Elgamal	ROM, DL	standard	statistical	uniform
[FL16]	Elgamal	knowledge	culpable	statistical	structured
[FLZ16]	ILin (lifted)	GGM	standard	statistical	structured
[GR16]	Elgamal	falsifiable	standard	computational	structured
[FFHR19]	pv-RRCCA	SXDH	computational UC-secure mix-net		uniform
$\mathcal{S}$ (Chapter 3)	Elgamal	GGM	standard	statistical	structured
$\mathcal{S}^+$ (Chapter 5)	Elgamal	AGM	standard	subversion statistical	structured

2. PRELIMINARIES

We start by introducing some basic notation used throughout this thesis. We denote the security parameter by λ. A function $f : \mathbb{N} \to [0,1]$ is called negligible if $f(\lambda) = O(\lambda^{-c})$ for every constant $c \in \mathbb{N}$. We denote this by $f \approx_\lambda 0$.

Integers modulo p are denoted by $\mathbb{Z}_p$ and the ring of polynomials with variables $X_1, \ldots, X_n$ over $\mathbb{Z}_p$ by $\mathbb{Z}_p[X_1, \ldots, X_n]$. Vectors $\mathbf{a} := (a_i)_{i=1}^n \in \mathbb{Z}_p^n$ are column vectors by default and are usually denoted by bold lower-case letters. Length n vector of ones is denoted by $\mathbf{1}_n$ and vector of zeros by $\mathbf{0}_n$. Matrices $\mathbf{A} \in \mathbb{Z}_p^{n \times m}$ are denoted bold upper-case letters and i-th row vector of $\mathbf{A}$ is denoted by $\mathbf{A}_i$ and i-th column by $\mathbf{A}^{(i)}$. Identity matrix is denoted by $\mathbf{I}_n \in \mathbb{Z}_p^{n \times n}$. Set of permutations on n elements is denoted by $\mathbb{S}_n$. Set $\{1, \ldots, n\}$ is denoted by $[1..n]$.

Sampling x from a distribution $\mathscr{D}$ is written as $x \leftarrow_{\$} \mathscr{D}$. If $\mathscr{D}$ is a set, then $x \leftarrow_{\$} \mathscr{D}$ means uniform sampling. We often express probabilities in the form $\Pr[x \leftarrow_{\$} \mathscr{D} : \mathsf{Pred}(x)]$ where $\mathscr{D}$ is a probability distribution and Pred some predicate. Sometimes we need to explicitly use random coins of an algorithm A. We denote sampling sufficiently long randomness by $r \leftarrow_{\$} \mathsf{RND}_\lambda(\mathsf{A})$ and A outputting y on input x and random coins r is denoted by $y \leftarrow \mathsf{A}(x; r)$. Often we leave out the random coins and just write $y \leftarrow \mathsf{A}(x)$. Probabilistic polynomial time is abbreviated by PPT. We assume that algorithms are stateful, e.g., we may first execute A on input x and then again on another input y without A losing its state from the first execution.

Lagrange polynomials. Let $\omega_1, \ldots, \omega_{n+1}$ be unique elements in $\mathbb{Z}_p$. Lagrange polynomial $\ell_i(X)$ is the unique n-th degree polynomial such that $\ell_i(\omega_i) = 1$ and $\ell_i(\omega_j) = 0$ for $j \neq i$. More explicitly we may express $\ell_i(X) = \prod_{j \neq i}^{n+1} \frac{X - \omega_i}{\omega_j - \omega_i}$. Lagrange polynomials $\ell_1(X), \ldots, \ell_{n+1}(X)$ form a basis for all polynomials in $\mathbb{Z}_p[X]$ which are at most degree n. Lagrange polynomials are often used for interpolation. Namely, if $f \in \mathbb{Z}_p[X]$ has a degree at most n and $f(\omega_i) = a_i$ for $i \in [1..n+1]$, then $f(X) = \sum_{i=1}^{n+1} a_i \ell_i(X)$.

2.1. Pairings

All the main constructions of this thesis use cryptographic bilinear pairings. Let BPgen be a determinstic polynomial time algorithm that on input 1^λ outputs a tuple $\mathsf{bp} = (p, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathscr{P}_1, \mathscr{P}_2)$ where p is a prime of length $\Theta(\lambda)$, $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are additive groups of order p with an efficient operation, $\mathscr{P}_1$ and $\mathscr{P}_2$ are respectively the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, and $\hat{e}$ is a map $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

Bilinearity: $\hat{e}(x\mathscr{P}_1, y\mathscr{P}_2) = (xy)\mathscr{P}_T$ where $\mathscr{P}_T := \hat{e}(\mathscr{P}_1, \mathscr{P}_2)$ for any $x, y \in \mathbb{Z}_p$.

Non-degeneracy: $\mathcal{P}_T$ as defined above is a non-zero element and thus also a generator of the target group $\mathbb{G}_T$.

Efficiency: $\hat{e}$ can be efficiently computed.

Pairings are usually divided into three types: Type 1 pairings have $\mathbb{G}_1 = \mathbb{G}_2$, Type 2 pairings have $\mathbb{G}_1 \neq \mathbb{G}_2$, and Type 3 pairings have $\mathbb{G}_1 \neq \mathbb{G}_2$ even such that there is no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. Type 3 pairings are currently the most widely used due to their efficiency and also in this thesis, we will use Type 3 pairings. We will not discuss how pairings are constructed and will use them as a black box. For state-of-the-art pairings, we refer the reader to [BD19].

Bracket Notation. We will be using pairings quite extensively and thus we are going to use convenient bracket notation of Escala et al. [EHK+13] which has become widely used in pairing-based research. Firstly, instead of writing $x\mathcal{P}_i$ for $i \in \{1, 2, T\}$, we write $[x]_i$. This extends naturally to matrices $[\mathbf{A}]_i \in \mathbb{G}_i^{n\times m}$ where $\mathbf{A} \in \mathbb{Z}_p^{n\times m}$. Secondly, instead of writing the pairing operation as $\hat{e}([x]_1, [y]_1)$, we write $[x]_1 \cdot [y]_2 = [x]_1[y]_2 = [xy]_T$ and similarly we can write $[\mathbf{A}]_1 \cdot [\mathbf{B}]_2 = [\mathbf{A}]_1[\mathbf{B}]_2 = [\mathbf{AB}]_T$ for matrices $\mathbf{A}$ and $\mathbf{B}$ of suitable dimensions. Sometimes we also write $[x]_{1,2}$ as a shorthand for $[x]_1$, $[x]_2$.

2.2. Assumptions

Let GPgen be a PPT group generator that on input 1^λ outputs $(p, \mathbb{G}, \mathcal{P})$ where $\mathbb{G}$ is an additive group of prime order $p = \Theta(\lambda)$ and $\mathcal{P}$ a random generator of the group. We still use the bracket notation to represent group elements but drop the lower index.

We start by defining some common assumptions in group-based cryptography.

Definition 1 (DL Assumption)**.** *Discrete Logarithm (DL) assumption holds relative to* GPgen*, if for any PPT adversary* A*,* $\varepsilon \approx_\lambda 0$*, where*

$$\varepsilon := \Pr\left[\ \mathsf{gp} \leftarrow \mathsf{GPgen}(1^\lambda), x \leftarrow_\$ \mathbb{Z}_p : \mathsf{A}(\mathsf{gp}, [x]) = x\ \right].$$

Definition 2 (q-PDL Assumption)**.** *q-Power Discrete Logarithm (PDL) assumption holds relative to* GPgen*, if for any PPT adversary* A*,* $\varepsilon \approx_\lambda 0$*, where*

$$\varepsilon := \Pr\left[\ \mathsf{gp} \leftarrow \mathsf{GPgen}(1^\lambda), x \leftarrow_\$ \mathbb{Z}_p : \mathsf{A}(\mathsf{gp}, [x, \ldots, x^q]) = x\ \right].$$

We can also have a similar assumption in the pairing-based setting.

Definition 3 (q-SPDL Assumption)**.** *q-Symmetric Power Discrete Logarithm (SPDL) assumption holds relative to* BPgen*, if for any PPT adversary* A*,* $\varepsilon \approx_\lambda 0$*, where*

$$\varepsilon := \Pr\left[\ \mathsf{bp} \leftarrow \mathsf{BPgen}(1^\lambda), x \leftarrow_\$ \mathbb{Z}_p : \mathsf{A}(\mathsf{gp}, [x, \ldots, x^q]_{1,2}) = x\ \right].$$

Definition 4 (CDH Assumption)**.** *Computational Diffie-Hellman (CDH) assumption holds relative to* GPgen*, if for any PPT adversary* A*,* $\varepsilon \approx_\lambda 0$*, where*

$$\varepsilon := \Pr\left[\ \mathsf{gp} \leftarrow \mathsf{GPgen}(1^\lambda), x, y \leftarrow_\$ \mathbb{Z}_p : \mathsf{A}(\mathsf{gp}, [x], [y]) = [xy]\ \right].$$

Definition 5 (DDH Assumption)**.** *Decisional Diffie-Hellman (DDH) assumption holds relative to* GPgen*, if for any PPT adversary* A*,* $|\varepsilon_0 - \varepsilon_1| \approx_\lambda 0$*, where*

$$\varepsilon_b := \Pr\left[\begin{array}{l} \mathsf{gp} \leftarrow \mathsf{GPgen}(1^\lambda), x, y \leftarrow_\$ \mathbb{Z}_p, \textit{ if } b = 0 \textit{ then } z \leftarrow xy \textit{ else } z \leftarrow_\$ \mathbb{Z}_p : \\ \mathsf{A}(\mathsf{gp}, [x], [y], [z]) = 1 \end{array}\right].$$

Morillo et al. [MRV16] introduced the Kernel Matrix Diffie-Hellman (KerMDH) assumption which generalizes many different previously known computational assumptions. We present it here for bilinear groups. Let $\mathscr{D}_{\ell,k}$ be some matrix distribution over $\mathbb{Z}_p^{\ell \times k}$ where $\ell > k$.

Definition 6 (KerMDH Assumption)**.** $\mathscr{D}_{\ell,k}$*-KerMDH (Kernel Matrix Diffie-Hellman) assumption holds relative to* BPgen*, if for any PPT adversary* A*,* $\varepsilon \approx_\lambda 0$*, where*

$$\varepsilon := \Pr\left[\begin{array}{l} \mathsf{bp} \leftarrow \mathsf{BPgen}(1^\lambda), \boldsymbol{A} \leftarrow_\$ \mathscr{D}_{\ell,k}, [\boldsymbol{x}]_2 \leftarrow \mathsf{A}(\mathsf{bp}, [\boldsymbol{A}]_1) : \\ \boldsymbol{A}^\top \boldsymbol{x} = \boldsymbol{0} \wedge \boldsymbol{x} \neq \boldsymbol{0} \end{array}\right].$$

One usually takes $\ell = k + 1$ for best efficiency and writes $\mathscr{D}_k := \mathscr{D}_{k+1,k}$. Some common $\mathscr{D}_k$ families, that are conjectured to be secure, include

$$\mathscr{L}_k := \begin{pmatrix} a_1 & 0 & \dots & 0 \\ 0 & a_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_k \\ 1 & 1 & \dots & 1 \end{pmatrix}, \mathscr{C}_k := \begin{pmatrix} a_1 & 0 & \dots & 0 \\ 1 & a_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & a_k \\ 0 & \dots & \dots & 1 \end{pmatrix}, \mathscr{U}_k := \begin{pmatrix} a_{1,1} & \dots & a_{1,k} \\ a_{2,1} & \dots & a_{2,k} \\ \dots & \dots & \dots \\ a_{k+1,1} & \dots & a_{k+1,k} \end{pmatrix},$$

where $a_i, a_{ij} \leftarrow_\$ \mathbb{Z}_p$.

We will also be using a very simple knowledge assumption.

Definition 7 (BDH-KE [ABLZ17])**.** *Bilinear Diffie-Hellman Knowledge of Exponent assumption holds relative to* BPgen*, if for any PPT* A *exists a PPT extractor* Ext*, such that*

$$\Pr\left[\begin{array}{l} \mathsf{bp} \leftarrow \mathsf{BPgen}(1^\lambda), r \leftarrow_\$ \mathsf{RND}_\lambda(\mathsf{A}), ([x]_1, [x]_2) \leftarrow \mathsf{A}(\mathsf{bp}; r), \\ x' \leftarrow \mathsf{Ext}(\mathsf{bp}; r) : x \neq x' \end{array}\right] \approx_\lambda 0.$$

Generic Group Model. Generic (bilinear) group model (GGM) [Sho97,Mau05] defines a restricted mode of computation in a group-based setting. The idea is to allow an algorithm to only make group-based operations and keep the structure of the group itself hidden. In the context of this thesis, we consider natural operations of bilinear groups which are exponentiation, group addition, pairing, and equality check. One way to formalize GGM is by making operations through oracle calls where oracle only returns a new memory index of group element or a bit in case of equality test. We refrain from giving a full formal description and redirect the reader to [Mau05].

GGM has found two common applications in cryptography. One is to justify novel group-based security assumptions by showing that they hold at least respect to generic adversaries, e.g. [Sho97, BBG05]. Second is by directly proving the security of a cryptographic primitive, e.g [BFF+15, Gro16]. Security proofs in GGM are usually unconditional rather than reductions to some assumption. Like RO model, GGM (at least the Shoup's version) is known to fail in some contrived protocols [Den02].

Algebraic Group Model. More recently, Fuchsbauer, Kiltz, and Loss [FKL18] proposed a relaxation of GGM that they called algebraic group model (AGM). Here, the group structure is revealed to the adversary and the only restriction is that the adversary has to provide a linear representation together with each group element. More precisely, suppose that an algebraic adversary gets $[x_1, \ldots, x_n]$ and gp as an input. Then together with each group element $[y]$ that adversary outputs, it also has to output a representation $a_0, a_1, \ldots, a_n \in \mathbb{Z}_p$ such that $y = a_0 + \sum_{i=1}^{n} a_i x_i$. This model is slightly more realistic since attacks can take into account the concrete structure of the group. On the other hand, security proofs in AGM are typically reductions (usually to some flavor of DL assumption) unlike unconditional proofs in GGM. Known weaknesses of GGM like [Den02] do not carry over to AGM as was recently observed in [AHK20].

Falsifiable vs. Non-Falsifiable Assumptions. Naor [Nao03] introduced the classification of security assumptions based on the notion of falsifiability. This was later re-formulated by Gentry and Wichs [GW11] (see also the classification by Pass [Pas13]) to what is currently understood by a falsifiable assumption. Gentry and Wichs called an assumption falsifiable if it can be formulated as an interactive protocol between an efficient challenger and an adversary such that the challenger can verify if the adversary won the game. Most cryptographic assumptions, e.g. DL, DDH assumptions from above, are falsifiable.

However, some assumptions do not fit under this umbrella. Some assumptions require the challenger to have superpolynomial computational power (see for example one-more discrete logarithm assumption [BNPS03]) and some seemingly do not fit into the challenger-adversary framework at all. Of the latter type are all kinds of knowledge assumptions. For example Damgård's knowledge-of-exponent assumption [Dam92] states that for any PPT adversary A there exist a PPT extractor Ext such that if A on input $([1], [x])$ outputs $([z], [zx])$, then Ext, given the same random coins and input, can output z with an overwhelming probability. Idea is that essentially the only way to compute $([z], [zx])$ is by first computing z and then multiplying $([1], [x])$ with z. Then Ext can be seen as a slight modification of A's code which outputs z instead of $([z], [zx])$. BDH-KE assumption is a variation of this assumption.

A common sentiment is that non-falsifiable assumptions are less trustworthy due to difficulty in assessing their validity (see for example [Nao03, GK16]). There is even some concrete evidence that knowledge assumptions do not hold respect to arbitrary (efficiently sampleable) auxiliary input [BP15]. AGM and GGM are even stronger than concrete group-based (knowledge) assumptions. However, AGM and GGM are often used to justify novel group-based assumptions. In our opinion, if the novel assumption is essentially equivalent to the cryptographic primitive, it might be more reasonable to directly prove the primitive itself in AGM or GGM.

2.3. Cryptosystems

A public-key cryptosystem is a triple of PPT algorithms $(\mathsf{Kgen}, \mathsf{Enc}, \mathsf{Dec})$. Key generator Kgen takes in a security parameter 1^λ and outputs a public key pk and a secret key sk. Encryption algorithm Enc takes as input a public key pk and a message $\mathsf{m} \in \mathscr{M}$ where $\mathscr{M}$ is the message space. Decryption algorithm takes as input a secret key sk and a ciphertext c and outputs a message m.

Cryptosystem should be functional in the sense that for any key pair $(\mathsf{pk}, \mathsf{sk})$ and $\mathsf{m} \in \mathscr{M}$ we have $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{m})) = \mathsf{m}$. A common security requirement is INDistinguishability under Chosen Plaintext Attack (IND-CPA) which requires that ciphertexts are indistinguishable even for known plaintexts. More precisely, for any PPT adversary A, $|\varepsilon_0 - \varepsilon_1| \approx_\lambda 0$ where,

$$\varepsilon_b := \Pr\left[\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Kgen}(1^\lambda), (\mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathsf{A}(\mathsf{pk}) : \mathsf{A}(\mathsf{Enc}(\mathsf{pk}, \mathsf{m}_b)) = 1\ \right].$$

We call a group based cryptosystem lifted if the message space is $\mathscr{M} = \mathbb{Z}_p$ and decryption involves computing the discrete logarithm of $[\mathsf{m}]$ to recover m. This means that only decryption of short messages, say 30 or 40 bits, is possible.

Elgamal Cryptosystem. In this thesis, we only use the (non-lifted) Elgamal cryptosystem [ElG84] that we define next. Let us sample a group description $(p, \mathbb{G}, \mathscr{P}) \leftarrow \mathsf{GPgen}(1^\lambda)$. We set the message space $\mathscr{M} = \mathbb{G}$. Key generator Kgen outputs a random secret key $\mathsf{sk} \leftarrow_\$ \mathbb{Z}_p$ and sets the public key to $\mathsf{pk} \leftarrow [1, \mathsf{sk}]$. Encryption algorithm Enc on input $(\mathsf{pk}, \mathsf{m} = [m] \in \mathbb{G})$ samples $r \leftarrow_\$ \mathbb{Z}_p$ and outputs $[0, m] + r \cdot [1, \mathsf{sk}]$. Sometimes we write $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}; r)$ when r is sampled outside of the algorithm Enc. Decryption algorithm Dec takes as an input $(\mathsf{sk}, \mathsf{c} = [c_1, c_2])$ and outputs $[c_2] - \mathsf{sk}[c_1]$.

It can be verified with a direct substitution that the Elgamal cryptosystem is functional. It also satisfies IND-CPA security under the DDH assumption. Moreover, the Elgamal cryptosystem has the blinding property which allows rerandomizing ciphertexts. Namely, given an arbitrary ciphertext $\mathsf{c} = [r, m + r \cdot \mathsf{sk}]$, we can pick a new randomness $r' \leftarrow_\$ \mathbb{Z}_p$ and compute a new ciphertext $\mathsf{c}' \leftarrow ([r'] + [r], r'[sk] + [m + r \cdot \mathsf{sk}]) = \mathsf{Enc}(\mathsf{pk}, [m]; r + r')$. Ciphertext c' still decrypts to the same message $[m]$ but is now computationally indistinguishable from a ciphertext with any other message.

2.4. Non-Interactive Zero-Knowledge Arguments

Let $\mathscr{L}$ be a NP-language and $\mathscr{R}$ the corresponding binary relation of all pairs (x, w) where x that we call a statement is in $\mathscr{L}$ and w is any of the efficiently verifiable witnesses for x.

A non-interactive zero-knowledge (NIZK) argument $\mathscr{A}$ for a relation $\mathscr{R}$ is a tuple of following PPT algorithms:

- Parameter generator Pgen that takes as an input a security parameter 1^λ and outputs a setup parameters par. In our construction par is the bilinear group description.
- CRS generation algorithm Cgen that takes as an input par and outputs a CRS crs and a trapdoor td.
- Prover algorithm P that takes an input $(\mathsf{par}, \mathsf{crs}, \mathsf{x}, \mathsf{w})$ such that $(\mathsf{x}, \mathsf{w}) \in \mathscr{R}$ and outputs an argument π.
- Verifier algorithm V that takes an input $(\mathsf{par}, \mathsf{crs}, \mathsf{x}, \pi)$ and either rejects the argument by outputting 0 or accepts the argument by outputting 1.
- Simulator algorithm Sim that takes an input $(\mathsf{par}, \mathsf{crs}, \mathsf{x}, \mathsf{td})$ with $\mathsf{x} \in \mathscr{L}$ and outputs a forged argument π.

NIZK arguments must always satisfy some flavor of the properties, completeness, soundness, and zero-knowledge. Completeness establishes that the protocol is functional, i.e., if the CRS generator, the prover, and the verifier are all honest, then the proof will be accepted. Soundness captures a security property that it should be computationally hard to construct an acceptable argument for an invalid statement $\mathsf{x} \notin \mathscr{L}$. Zero-knowledge means that the algorithm Sim, given additionally a trapdoor td, can efficiently produce an argument π that is indistinguishable from an honest argument. Therefore if one knows the trapdoor, then he does not actually "learn" anything new from the proof besides that $x \in \mathscr{L}$. We define these properties more formally below.

Definition 8 (Completeness)**.** *An argument $\mathscr{A}$ is perfectly complete if for any $\lambda \in \mathbb{N}$ and $(\mathsf{x}, \mathsf{w}) \in \mathscr{R}$,*

$$\Pr\left[\begin{array}{l} \mathsf{par} \leftarrow \mathsf{Pgen}(1^\lambda), (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Cgen}(\mathsf{par}), \pi \leftarrow \mathsf{P}(\mathsf{par}, \mathsf{crs}, \mathsf{x}, \mathsf{w}) : \\ \mathsf{V}(\mathsf{par}, \mathsf{crs}, \mathsf{x}, \pi) = 1 \end{array}\right] = 1.$$

Definition 9 (Soundness)**.** *An argument $\mathscr{A}$ is computationally sound if for any PPT adversary* A*,*

$$\Pr\left[\begin{array}{l} \mathsf{par} \leftarrow \mathsf{Pgen}(1^\lambda), (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Cgen}(\mathsf{par}), (\mathsf{x}, \pi) \leftarrow \mathsf{A}(\mathsf{par}, \mathsf{crs}) : \\ \mathsf{V}(\mathsf{par}, \mathsf{crs}, \mathsf{x}, \pi) = 1 \wedge \mathsf{x} \notin \mathscr{L} \end{array}\right] \approx_\lambda 0.$$

One can strengthen the above definition and require that prover has knowledge of the witness w corresponding to x. By knowledge, we mean that an adversary could efficiently compute w. We capture this with an efficient extraction algorithm Ext that outputs w given the same input and the same random tape as the adversary.

Definition 10 (Knowledge Soundness)**.** *An argument $\mathscr{A}$ is computationally knowledge sound if for any PPT adversary* A*, there exists a PPT extractor* Ext *such that,*

$$\Pr\left[\begin{array}{l} \mathsf{par} \leftarrow \mathsf{Pgen}(1^\lambda), (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Cgen}(\mathsf{par}), r \leftarrow_\$ \mathsf{RND}_\lambda(\mathsf{A}), \\ (\mathsf{x}, \pi) \leftarrow \mathsf{A}(\mathsf{par}, \mathsf{crs}; r), \mathsf{w} \leftarrow \mathsf{Ext}(\mathsf{par}, \mathsf{crs}; r) : \\ \mathsf{V}(\mathsf{par}, \mathsf{crs}, \mathsf{x}, \pi) = 1 \wedge (\mathsf{x}, \mathsf{w}) \notin \mathscr{R} \end{array}\right] \approx_\lambda 0.$$

Groth, Ostrovsky, and Sahai [GOS06a, GOS06b] introduced a weaker form of soundness called culpable soundness (originally called co-soundness) which intuitively guarantees security only against adversaries that can provide a witness of their cheating. More formally, we fix an efficiently verifiable relation $\mathscr{R}^{glt} = \{(\mathsf{x}, \mathsf{w}^{glt}) : \mathsf{x} \in \bar{\mathscr{L}}\}$ where $\bar{\mathscr{L}}$ is the complement of the language $\mathscr{L}$ and define culpable soundness as follows.

Definition 11 (Culpable Soundness)**.** *An argument $\mathscr{A}$ is computationally culpably sound respect to $\mathscr{R}^{glt}$ if for any PPT adversary* A*,*

$$\Pr\left[\begin{array}{l} \mathsf{par} \leftarrow \mathsf{Pgen}(1^\lambda), (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Cgen}(\mathsf{par}), (\mathsf{x}, \mathsf{w}^{glt}, \pi) \leftarrow \mathsf{A}(\mathsf{par}, \mathsf{crs}) : \\ \mathsf{V}(\mathsf{par}, \mathsf{crs}, \mathsf{x}, \pi) = 1 \wedge (\mathsf{x}, \mathsf{w}^{glt}) \in \mathscr{R}^{glt} \end{array}\right] \approx_\lambda 0.$$

The idea here is that choosing a suitable $\mathscr{R}^{glt}$ can allow using w^{glt} to make a reduction to some security assumptions. Of course, in reality, there can be PPT adversaries that do not know w^{glt} but can still provide an acceptable proof π for a false statement x. Moreover, note that culpable soundness is possible only for languages in $\mathsf{NP} \cap \mathsf{coNP}$.

Definition 12 (Zero-Knowledge)**.** *An argument $\mathscr{A}$ is perfectly zero-knowledge if for any adversary* A*, $\varepsilon_0 = \varepsilon_1$, where*

$$\varepsilon_b := \Pr\left[\begin{array}{l} \mathsf{par} \leftarrow \mathsf{Pgen}(1^\lambda), (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Cgen}(\mathsf{par}), (\mathsf{x}, \mathsf{w}) \leftarrow \mathsf{A}(\mathsf{par}, \mathsf{crs}), \\ \textit{If } b = 0\textit{, then } \pi \leftarrow \mathsf{P}(\mathsf{par}, \mathsf{crs}, \mathsf{x}, \mathsf{w}) \textit{ else } \pi \leftarrow \mathsf{Sim}(\mathsf{par}, \mathsf{crs}, \mathsf{x}, \mathsf{td}) : \\ \mathsf{A}(\pi) = 1 \end{array}\right].$$

There is a slightly weaker form of privacy called witness indistinguishability (WI) which requires that adversary cannot distinguish which witness was used to construct the proof. Observe however that if a statement x has only one valid witness w, then WI can hold even if argument leaks w.

Definition 13 (Witness-Indistinguishability)**.** *An argument $\mathscr{A}$ is perfectly witness-indistinguishable if for any adversary* A*, $\varepsilon_0 = \varepsilon_1$, where*

$$\varepsilon_b := \Pr\left[\begin{array}{l} \mathsf{par} \leftarrow \mathsf{Pgen}(1^\lambda), (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Cgen}(\mathsf{par}), (\mathsf{x}, \mathsf{w}_0, \mathsf{w}_1) \leftarrow \mathsf{A}(\mathsf{par}, \mathsf{crs}), \\ \pi \leftarrow \mathsf{P}(\mathsf{par}, \mathsf{crs}, \mathsf{x}, \mathsf{w}_b) : \mathsf{A}(\pi_b) = 1 \wedge (\mathsf{x}, \mathsf{w}_0) \in \mathscr{R} \wedge (\mathsf{x}, \mathsf{w}_1) \in \mathscr{R} \end{array}\right].$$

Subversion zero-knowledge. Finally, we present a very strong notion of zero-knowledge where even the CRS generator can be malicious. We follow the definition from [ABLZ17]. The idea is to additionally have a CRS verification algorithm CV which is used by the prover. Prover reveals the proof only if CRS passes the verification. To simulate, we need to make sure that the CRS subverter knows the trapdoor in case the CRS passes verification. Thus, we also require an extractor algorithm that outputs a trapdoor given subverter's random coins as an input.

Formally, we say that a NIZK argument $\mathscr{A}$ for relation $\mathscr{R}$ is subversion resistant if it additionally contains the following PPT algorithm.

- CRS verification algorithm CV that takes as an input $(\mathsf{par}, \mathsf{crs})$ and either accepts the CRS by outputting 1 or rejects it by outputting 0.

Definition 14 (Sub-ZK)**.** *A subversion resistant argument $\mathscr{A}$ is statistically subversion zero-knowledge if for any PPT subverter* Sub*, there exists an extractor* Ext*, such that for any adversary* A*, $\varepsilon_0 \approx_\lambda \varepsilon_1$, where*

$$\varepsilon_b := \Pr\left[\begin{array}{l} \mathsf{par} \leftarrow \mathsf{Pgen}(1^\lambda), r \leftarrow_{\$} \mathsf{RND}_\lambda(\mathsf{Sub}), (\mathsf{crs}, \mathsf{aux}) \leftarrow \mathsf{Sub}(\mathsf{par}; r), \\ \mathsf{td} \leftarrow \mathsf{Ext}(\mathsf{par}; r), (\mathsf{x}, \mathsf{w}) \leftarrow \mathsf{A}(\mathsf{aux}), \\ \textit{If } b = 0\textit{, then } \pi \leftarrow \mathsf{P}(\mathsf{par}, \mathsf{crs}, \mathsf{x}, \mathsf{w}) \textit{ else } \pi \leftarrow \mathsf{Sim}(\mathsf{par}, \mathsf{crs}, \mathsf{x}, \mathsf{td}) : \\ \mathsf{CV}(\mathsf{par}, \mathsf{crs}) = 1 \wedge \mathsf{A}(\pi) = 1 \end{array}\right].$$

2.5. Universal Composability

Universal composability (UC) [Can01] is a security model where protocols retain their security properties even when arbitrarily composed with other protocols or different instances of the same protocol. Security requirements in the UC model are captured by an ideal functionality. We have two games called the real world (includes an actual protocol) and an ideal world (with an ideal functionality) that should be indistinguishable. Below we give their rough description, but for a detailed explanation, we refer the reader to the original paper by Canetti [Can01].

The real-world game $\mathsf{Real}_{\mathsf{Z},\mathsf{A},\Pi}$ consists of PPT algorithms: environment Z, adversary A, some number of parties $\mathsf{P}_1, \dots, \mathsf{P}_N$, and a protocol Π. Environment Z models all the other possible protocols running in parallel. The environment is allowed to give inputs and receive outputs from parties and freely communicate with A. Adversary A can corrupt new parties at any point in the execution and will have full control over them which includes seeing their internal state. Parties in real-world execute the protocol Π and the communication is routed through A. In the end, Z will output a guess 1 if it believes to be in the real world and 0 otherwise.

In the ideal world game $\mathsf{Ideal}_{\mathsf{Z},\mathsf{Sim},\mathsf{F}}$ we also have an environment Z and parties $\mathsf{P}_1, \dots, \mathsf{P}_N$ but instead of the adversary we have a simulator Sim and instead of Π we have an ideal functionality F. Ideal functionality is essentially a trusted third party that receives input data (possibly private data) from parties and returns them the correct output. Depending on the primitive there can even be back and forth communication between F and the parties. (See DL-extractable commitment in Section 2.6 for a relatively simple example.) Parties in the ideal world only transfer inputs to the ideal functionality and outputs from ideal functionality back to the environment. Simulator Sim has to be able to simulate the output of A which might include for example the protocol transcript. Importantly, Sim does not see the communication between ideal functionality and parties. Finally, Z again outputs a guess.

Definition 15. *We say that a protocol* Π *UC-realizes a functionality* F *if for any PPT* A*, there exists a PPT simulator* Sim *such for all PPT* Z *and $x \in \{0,1\}^{\mathsf{poly}(\lambda)}$,* $\Pr[\mathsf{Real}_{\mathsf{Z},\mathsf{A},\Pi}(x) = 1] \approx_\lambda \Pr[\mathsf{Ideal}_{\mathsf{Z},\mathsf{Sim},\mathsf{F}}(x) = 1]$ *where x is an auxiliary input to* Z.

If Π *uses some ideal functionality* F^* *as a subprotocol to UC-realize* F*, then we say that* Π *UC-realizes* F *in* F^**-hybrid model.*

The main importance of the UC model is the universal composability theorem which, roughly speaking, says that Π that UC-realizes a functionality is secure even when running concurrently with other arbitrary protocols. See [Can01] for the formally rigorous statement.

2.6. Commitment Schemes

A commitment scheme is a pair of PPT algorithms $\mathscr{C} = (\mathsf{Kgen}, \mathsf{Com})$. Key generation algorithm Kgen takes in a security parameter 1^λ and outputs a commitment key ck. Commitment algorithm Com takes in ck, a message m from message space $\mathscr{M}$, randomness r from randomness space $\mathscr{R}$ and outputs a commitment c. We can think of a commitment scheme as a cryptosystem which does not necessarily have a decryption algorithm. Commitment c should not reveal the message m (hiding) and it should be computationally difficult to find $\mathsf{r}' \in \mathscr{R}$ and $\mathsf{m}' \in \mathscr{M}$ such that $\mathsf{c} = \mathsf{Com}(\mathsf{ck}, \mathsf{m}'; r)$ but $\mathsf{m} \neq \mathsf{m}'$ (binding).

Definition 16 (hiding)**.** *A commitment scheme* $\mathscr{C}$ *is computationally hiding if for any PPT adversary* A*,* $|\varepsilon_0 - \varepsilon_1| \approx_\lambda 0$ *where,*

$$\varepsilon_b := \Pr\left[\ \mathsf{ck} \leftarrow \mathsf{Kgen}(1^\lambda), (\mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathsf{A}(\mathsf{ck}) : \mathsf{A}(\mathsf{Com}(\mathsf{ck}, \mathsf{m}_b)) = 1\ \right].$$

$\mathscr{C}$ *is perfectly hiding if* $\varepsilon_0 = \varepsilon_1$ *for even any computationally unbounded* A.

Definition 17 (binding)**.** *A commitment scheme* $\mathscr{C}$ *is computationally binding if for any PPT adversary* A*,* $\varepsilon \approx_\lambda 0$ *where,*

$$\varepsilon := \Pr\left[\begin{array}{l} \mathsf{ck} \leftarrow \mathsf{Kgen}(1^\lambda), (\mathsf{m}_0, \mathsf{m}_1, \mathsf{r}_0, \mathsf{r}_1) \leftarrow \mathsf{A}(\mathsf{ck}) : \\ \mathsf{Com}(\mathsf{ck}, \mathsf{m}_0; \mathsf{r}_0) = \mathsf{Com}(\mathsf{ck}, \mathsf{m}_1; \mathsf{r}_1) \wedge \mathsf{m}_0 \neq \mathsf{m}_1 \end{array}\right].$$

$\mathscr{C}$ *is perfectly binding if* $\varepsilon = 0$ *for even any unbounded* A.

Every cryptosystem can be used as a perfectly binding and computationally hiding commitment scheme.

p(X)*-Multi-Pedersen Commitment.* We formalize a slight extension of the common (Multi-)Pedersen commitment. Let $\mathbb{G}$ be a cyclic group of prime order p as before. We define the message space as $\mathbb{Z}_p^n$ and randomness space as $\mathbb{Z}_p$. Let $\mathbf{p}(X) = (p_1(X), \dots, p_n(X))^\top$ be a vector of linearly independent polynomials of (small) degree at most d. We define the $\mathbf{p}(X)$-Multi-Pedersen (MP) commitment scheme as follows. Key generator Kgen samples $x, \rho \leftarrow_\$ \mathbb{Z}_p$ such that $\rho \neq 0$ and outputs $\mathsf{ck} \leftarrow [\rho, p_1(x), \dots, p_n(x)]$. Commitment algorithm Com takes in ck, $\mathbf{m} \in \mathbb{Z}_p^n$, $r \leftarrow_\$ \mathbb{Z}_p$ and outputs $\mathsf{c} \leftarrow r[\rho] + \sum_{i=1}^{n} m_i [p_i(x)]$.

Theorem 1. ***p***(X)*-MP is perfectly hiding and computationally binding under d-PDL assumption.*

Let parties be $\mathsf{P}_1, \ldots, \mathsf{P}_N$ and the message space is $\mathbb{Z}_p$.

- On receiving input $(\mathsf{commit}, \mathsf{sid}, \mathsf{cid}, \mathsf{P}_i, \mathsf{P}_j, m)$ from party P_i where $m \in \mathbb{Z}_p$, if no tuple with $(\mathsf{sid}, \mathsf{cid})$ has been recorded, then store $(\mathsf{commit}, \mathsf{sid}, \mathsf{cid}, \mathsf{P}_i, \mathsf{P}_j, m)$ and send $(\mathsf{received}, \mathsf{sid}, \mathsf{cid}, \mathsf{P}_i, \mathsf{P}_j)$ to P_j and simulator Sim.
- On receiving $(\mathsf{open}, \mathsf{sid}, \mathsf{cid})$ from P_i, if a tuple $(\mathsf{commit}, \mathsf{sid}, \mathsf{cid}, \mathsf{P}_i, \mathsf{P}_j, m)$ has been recorded then send $(\mathsf{open}, \mathsf{sid}, \mathsf{cid}, \mathsf{P}_i, \mathsf{P}_j, [m])$ to P_j and Sim.

Figure 1. Ideal functionality $\mathsf{F}_{\mathit{dl\text{-}com}}$ for DL-extractable commitment

Proof. Hiding. Since $[\rho] \neq [0]$ is a generator of $\mathbb{G}$, then $r[\rho]$ is a uniformly random in $\mathbb{G}$ and thus the commitment itself is uniformly random and indepenent of the message $\mathbf{m}$. It follows that the scheme is perfectly hiding.

Binding. Suppose a PPT adversary A outputs $(\mathbf{m}, r, \mathbf{m}', r')$ with probability ε such that $r[\rho] + [\sum_{i=1}^{n} m_i p_i(x)] = r'[\rho] + [\sum_{i=1}^{n} m'_i p_i(x)]$ but $\mathbf{m} \neq \mathbf{m}'$. We will construct an adversary B that breaks the d-PDL assumption with the same probability. First, B receives $[x, \ldots, x^d]$ as a challenge and samples $\rho \leftarrow_{\$} \mathbb{Z}_p$ which is enough (together with the input) to construct the commitment key ck. It runs A on input ck to get $(\mathbf{m}, r, \mathbf{m}', r')$. Let us define a polynomial $p(X) := \sum_{i=1}^{n} (m_i - m'_i) p_i(X)$. Observe that if A was successful, then $p(X)$ is a non-zero polynomial since $\{p_i(X)\}_i$ are linearly independent and at least one of the coefficient $m_i - m'_i$ is non-zero. Thus, B can run an efficient root-finding algorithm to obtain $d+1$ roots of $p(X)$ (assuming that d is small), check which of them is x, and then return it. □

This commitment scheme is also additively homomorphic which means that $\mathsf{Com}(\mathsf{ck}, \mathbf{m}_1; r_1) + \mathsf{Com}(\mathsf{ck}, \mathbf{m}_2; r_2) = \mathsf{Com}(\mathsf{ck}, \mathbf{m}_1 + \mathbf{m}_2; r_1 + r_2)$.

DL-extractable Commitment. We also use a discrete logarithm (DL) extractable commitment from [ABL+19a]. DL-extractable commitment allows us to commit to a message $m \in \mathbb{Z}_p$, but only $[m]$ needs to be revealed for opening. However, it should be possible to extract m from the commitment when given a secret key.

Abdolmaleki et al. [ABL+19a] gave a UC-functionality of Fig. 1 for this primitive. There are N parties. Different execution sessions are differentiated by unique session IDs sid and different commitments in the same session are differentiated by unique commitment IDs cid. For committing, sending party will send a message m and recipient information to the functionality, upon which the functionality informs simulator and receiver that a commitment was made. Later sender can send an open message to the functionality and functionality will send $[m]$ to the simulator and the receiver. Such a commitment must necessarily be extractable. Namely, the only way to simulate a malicious commitment is for Sim to extract m from the commitment and send it to the ideal functionality.

Abdolmaleki et al. proposed an efficient construction in the registered public key (RPK) model [BCNP04] where, roughly speaking, each party can have their own trusted authority that stores the secret key. This is a somewhat weaker model than the CRS model. Moreover, it is known that every UC-functionality can be achieved in the uniform random string model [CLOS02] where all parties have access to a common uniformly random string. This construction would not be as efficient but would still serve our concrete application as we only need to use the commitment a small number of times.

3. AN EFFICIENT SHUFFLE ARGUMENT

3.1. Motivation

As we discussed in the introduction, zero-knowledge shuffle arguments allow us to construct verifiable mix-networks. Mix-nets essentially form a hard-to-trace communication channel similar to well-known Tor network [TOR18] but with slightly different properties. Namely, mix-network has provable security if at least one mixer is honest but overall lower latency. This makes it more suitable for applications like ballot shuffling in i-voting and anonymous e-mail [Cha81] and less suitable for anonymous instant messaging. For a longer overview of different types of mix-networks and their applications, we refer the reader to [SP06].

Our goal with this work is to come up with a secure and efficient shuffle argument. We i-voting in mind as our main application which can require hundreds of thousands or even millions of ciphertexts to be shuffled. As we mentioned previously, many efficient interactive shuffle arguments are known and they can be transformed into non-interactive arguments in the RO model. However, the RO model is flawed (at least) on a theoretical level and thus we are going to study shuffle arguments that are RO-free.

3.2. Overview

We construct a non-interactive zero-knowledge shuffle argument $\mathcal{S}$ in the pairing-based setting. Let $\mathsf{pk} = [1, \mathsf{sk}]_1$ be an Elgamal public key[1] and n the number of input ciphertexts. We represent n input and output ciphertexts respectively as matrices $[\mathbf{C}]_1, [\mathbf{C}']_1 \in \mathbb{G}_2^{n \times 2}$ where each row $[\mathbf{C}_i]_1$ and $[\mathbf{C}'_i]_1$ is a ciphertext. Ciphertexts are shuffled correctly if there exists a permutation $\sigma \in \mathbb{S}_n$ and randomizers $\mathbf{t} \in \mathbb{Z}_p^n$ such that $[\mathbf{C}_i]_1 = [\mathbf{C}_{\sigma(i)}]_1 + \mathsf{Enc}(\mathsf{pk}, [0]_1; t_i)$ for $i = 1, \dots, n$. If we represent the permutation σ as a permutation matrix $\mathbf{A}$, i.e., $\mathbf{A} = (a_{i,j}) \in \{0,1\}^{n \times n}$ such that $a_{i,j} = 1$ if and only if $\sigma(i) = j$, then the shuffle relation is as follows

$$\mathscr{R}_{sh}^{(n)} = \left\{ \begin{array}{l} \big(([\mathbf{C}, \mathbf{C}']_1, (\mathbf{A}, \mathbf{t})\big) \in \mathbb{G}_1^{n \times 4} \times (\mathbb{Z}_p^{n \times n} \times \mathbb{Z}_p^n) \mid \\ \mathbf{C}' = \mathbf{A} \cdot \mathbf{C} + \mathbf{t} \cdot \mathsf{pk} \wedge \mathbf{A} \text{ is a permutation matrix} \end{array} \right\}. \tag{3.1}$$

Essential to our construction is a $\mathbf{p}(X)$-Multi-Pedersen commitment with certain well-chosen polynomials $\mathbf{p}(X)$. The main idea is to commit to each column of the matrix $\mathbf{A}$ and to randomness vector $\mathbf{t}$, and then (i) prove knowledge of the permutation matrix $\mathbf{A}$, (ii) prove that the committed permutation and $\mathbf{t}$ were used to shuffle the ciphertexts. The first argument we call a permutation matrix argument and the second we call a consistency argument.

Unfortunately, we are unable to make both arguments work respect to the same $\mathbf{p}(X)$-MP commitments. Thus we commit once more to each row of $\mathbf{A}$ but this

[1] We present version of the argument where cryptosystem uses $\mathbb{G}_1$. Original paper [FLSZ17a] gave a construction where ciphertexts are in $\mathbb{G}_2$ which is slightly less efficient.

$\mathsf{Cgen}_{sm}(\mathsf{par},[\mathbf{M}]_2)$: $\mathbf{K} \leftarrow_{\$} \mathbb{Z}_p^{2\times 1}$; $[\mathbf{P}]_2 \leftarrow [\mathbf{M}]_2^\top \mathbf{K} \in \mathbb{Z}_p^{n\times 1}$; Return $(\mathsf{crs} = ([\mathbf{K}]_1,[\mathbf{P}]_2), \mathsf{td} = \mathbf{K})$;

$\mathsf{P}_{sm}(\mathsf{par},\mathsf{crs},\mathsf{x} = [a,\hat{a}]_2, w = (m_1,\dots,m_n,r,\hat{r}))$: Return $[d]_2 \leftarrow w[\mathbf{P}]_2$;

$\mathsf{V}_{sm}(\mathsf{par},\mathsf{crs},\mathsf{x} = [a,\hat{a}]_2, \pi = [d]_2)$: $[1]_1[d]_2 \stackrel{?}{=} [\mathbf{K}]_1^\top [a,\hat{a}]_2^\top$;

Figure 2. Same-message argument

time with a different $\mathbf{q}(X)$-MP commitment scheme (polynomials $\mathbf{p}(X)$ and $\mathbf{q}(X)$ are defined later). Finally, we construct a same-message argument which shows that prover knows how to open both commitments to the same vector. Below we give a more detailed explanation of each sub-argument. We state the main theorems, but for proofs refer the reader to [FLSZ17b].

We note that the conference version [FLSZ17a] contained a subtle flaw. Original construction used the same randomness for $\mathbf{p}(X)$-MP and $\mathbf{q}(X)$-MP commitments. This turned out to be insecure and we fixed it later in the full version [FLSZ17b].

3.3. Same-Message Argument

Let $\mathsf{ck} = [\rho, p_1(x),\dots,p_n(x)]_2$ and $\hat{\mathsf{ck}} = [\hat{\rho}, q_1(x),\dots,q_n(x)]_2$ be respectively commitment keys for $\mathbf{p}(X)$-MP commitment scheme and $\mathbf{q}(X)$-MP commitment scheme (we will use specific polynomials in later arguments). We want to prove knowledge of an opening of two commitments $[a]_2$ and $[\hat{a}]_2$ which use the respective commitment schemes. More concretely,

$$\mathscr{R}_{sm}^{(n)} = \left\{ \begin{array}{l} ([a,\hat{a}]_2^\top,(m_1,\dots,m_n,r,\hat{r})) \in \mathbb{G}_2^2 \times \mathbb{Z}_p^{n+2} \mid a = r\cdot\rho + \sum_{i=1}^n m_i p_i(x) \\ \wedge \hat{a} = \hat{r}\cdot\hat{\rho} + \sum_{i=1}^n m_i q_i(x) \end{array} \right\}.$$

Let us first observe that the above language can be restated as a linear language $[a,\hat{a}]_2^\top = [\mathbf{M}]_2 \cdot w^\top$ where

$$\mathbf{M} = \begin{pmatrix} \mathbf{p}(x) & \rho & 0 \\ \mathbf{q}(x) & 0 & \hat{\rho} \end{pmatrix} \in \mathbb{Z}_p^{2\times(n+2)} \text{ and } w = (m_1,\dots,m_n,r,\hat{r}).$$

Kiltz and Wee [KW15] proposed an efficient quasi-adaptive[2] NIZK argument for linear subspace languages where in the most efficient setting argument size is only 1 group element. Moreover, zero-knowledge is perfect and soundness holds under a standard falsifiable assumption. In Fig. 2 we have (the most efficient version of) Kiltz-Wee linear subspace argument adapted to our concrete matrix $\mathbf{M}$. We

[2]Quasi-adaptive means that CRS can depend on the language parameter, in our case $[\mathbf{M}]_2$.

```
$\mathsf{Cgen}_{uv}(\mathsf{par})$:
    1. $x, \rho \leftarrow_{\$} \mathbb{Z}_p$;
    2. $\mathbf{u} \leftarrow \{p_i(x)\}_{i=0}^{n} \cup \{\rho\}$; $\mathbf{v} \leftarrow \{\big((p_i(x)+p_0(x))^2-1\big)/\rho\}_{i=1}^{n}$;
    3. Return $(\mathsf{crs} = ([\mathbf{u}]_1, [\mathbf{u},\mathbf{v}]_2), \mathsf{td} = (x, \rho))$;
$\mathsf{P}_{uv}(\mathsf{par}, \mathsf{crs}, \mathsf{x} = [a]_2, w = (I, r))$:
    1. $[b]_1 \leftarrow [p_I(x)]_1 + r[\rho]_1$;
    2. $[c]_2 \leftarrow 2r[a]_2 + 2r[p_0]_2 - r^2[\rho]_2 + [((p_I(x)+p_0(x))^2-1)/\rho]_2$;
    3. Return $\pi = ([b]_1, [c]_2)$;
$\mathsf{V}_{uv}(\mathsf{par}, \mathsf{crs}, \mathsf{x} = [a]_2, \pi = ([b]_1, [c]_2))$:
    1. $\alpha \leftarrow_{\$} \mathbb{Z}_p$;
    2. $([b]_1 + [\alpha]_1 + [p_0(x)]_1)([a]_2 - [\alpha]_2 + [p_0(x)]_2) \overset{?}{=} [\rho]_1[c]_2 + [1-\alpha^2]_T$;
```

Figure 3. Unit vector argument

additionally need the same message argument to be knowledge sound. We prove this in GGM.

Theorem 2. *The same-message argument is perfectly complete, knowledge sound in GGM, and perfectly zero-knowledge.*

3.4. Unit Vector Argument

With a unit vector argument, we want to prove that prover knows a unit vector opening to some $\mathbf{p}(X)$-MP commitment $[a]_2$. More formally, we need a knowledge sound argument for the following relation

$$\mathscr{R}_{uv}^{(n)} = \left\{ \ ([a]_2, (\mathbf{m}, r)) \in \mathbb{G}_2 \times \mathbb{Z}_p^{n+1} \mid a = r \cdot \rho + \sum_{i=1}^{n} m_i p_i(x) \wedge \mathbf{m} \in \mathscr{U}_n \ \right\},$$

where $\mathscr{U}_n$ is the set of all unit vectors of length n, i.e, the set of binary vectors $\{0,1\}^n$ where exactly one position is 1. Alternatively, we can represent the witness as $(I, r) \in [1..n] \times \mathbb{Z}_p$ where I indicates that $m_I = 1$. This allows us to express $a = r \cdot \rho + p_I(x)$ in the previous relation.

Similarly to [FL16,FLZ16], we construct the unit vector argument using square span programs of Danezis et al. [DFGK14]. Vector $\mathbf{m}$ is a unit vector if $m_i \in \{0,1\}$ for $i \in [1..n]$ and $\sum_{i=1}^{n} m_i = 1$. We can restate this as the following matrix equation,

$$(\mathbf{V}\mathbf{m} + \mathbf{b} - \mathbf{1}_{n+1}) \circ (\mathbf{V}\mathbf{m} + \mathbf{b} - \mathbf{1}_{n+1}) = \mathbf{1}_{n+1}, \qquad (3.2)$$

where $\mathbf{V} = \begin{pmatrix} 2 \cdot \mathbf{I}_n \\ \mathbf{1}_n^\top \end{pmatrix}$, $\mathbf{b} = \begin{pmatrix} \mathbf{0}_n \\ 1 \end{pmatrix}$, and $\circ$ denotes point-wise multiplication. First n rows assert that $(2m_i - 1)^2 = 1$ which is equivalent to $m_i(m_i - 1) = 0$ and holds if and only if $m_i \in \{0,1\}$. Last row of the equation shows that $(\sum_{i=1}^{n} m_i)^2 = 1$ which is equivalent to $(\sum_{i=1}^{n} m_i - 1)(\sum_{i=1}^{n} m_i + 1) = 0$. Assuming that $n < p-1$ and $\mathbf{m} \in \{0,1\}^n$, the only solution is $\sum_{i=1}^{n} m_i = 1$. Thus, Eq. (3.2) is satisfied when $\mathbf{m}$ is a unit vector.

We will use this equation to construct a zk-SNARK for the unit vector relation. Let us fix polynomials $\mathbf{p}(x)$ in a particular way to enable this. We define $p_i(X)$ as an interpolation polynomial of the i-th column of $\mathbf{V}$, that is,

$$p_i(X) := 2\ell_i(X) + \ell_{n+1}(X) \text{ for } i \in [1..n],$$

where $\ell_i(X)$ is the i-th Lagrange polynomial. Additionally we define $p_0(X)$ as an interpolation of $\mathbf{b} - \mathbf{1}_{n+1}$,

$$p_0(X) := -\sum_{i=1}^{n} \ell_i(X) = 1 - \ell_{n+1}(X).$$

Last equality holds since interpolation of the constant polynomial 1 is equal to $\sum_{i=1}^{n+1} \ell_i(X)$.

Using our newly defined polynomials, we can state Eq. (3.2) in a polynomial form which is better suited for a zk-SNARK. Let us define $s(X) := (\sum_{i=1}^{n} m_i p_i(X) + p_0(X))^2 - 1$ and $t(X) := \prod_{i=1}^{n+1} X - \omega_i$, where ω_i are interpolation points of Lagrange polynomials (see Chapter 2). It is easy to verify that if $s(\omega_i) = 0$ for $i \in [1..n+1]$ then Eq. (3.2) is satisfied, or alternatively, if $t(X)$ divides $s(X)$ then Eq. (3.2) is satisfied.

Now we are ready to give an intuition for the unit vector argument in Fig. 3. The statement of the language is $[a]_2 = [p_I(x)]_2 + r[\rho]_2$. Prover outputs as part of the proof the same commitment also in the first group $[b]_1 = [p_I(x)]_1 + r[\rho]_1$. In particular, this means that we need commitment key of $\mathbf{p}(X)$-MP commitment as part of the CRS and key is given both in $\mathbb{G}_1$ and $\mathbb{G}_2$. We verify the equality of commitments with an equation $[b]_1[1]_2 = [1]_1[a]_2$. To verify that $t(X) \mid s(X)$, we proceed as follows. We include in the CRS elements $[((p_i(x) + p_0(x))^2 - 1)/\rho]_2$ for $i \in [1..n]$ where division by ρ guarantees linear independence of other CRS elements. Prover sends I-th of those elements in the proof (element $[c]_2$ in *Fig.* 3) together with some extra payload to cancel out the randomness of commitments. Verifier will check that $([b]_1 + [p_0(x)]_1)([a]_2 + [p_0(x)]_2) - [1]_T = [\rho]_1[c]_2$ where the left-hand side essentially corresponds to $s(X)$ and the right-hand side must be (due to multiplication by ρ) in the span of $\{(p_i(X) + p_0(X))^2 - 1\}$ which is

```
$\mathsf{Cgen}_{pm}(\mathsf{par})$: Return $\mathsf{Cgen}_{uv}(\mathsf{par})$;
$\mathsf{P}_{pm}(\mathsf{par}, \mathsf{crs}, \mathsf{x} = [\mathbf{a}]_2, w = (\mathbf{A}, \mathbf{r}))$:
    1. For $i \in [1..n]$: $\pi_i \leftarrow \mathsf{P}_{uv}(\mathsf{par}, \mathsf{crs}, [a_i]_2, (\mathbf{A}^{(i)}, r_i))$;
    2. Return $\pi = (\pi_1, \ldots, \pi_n)$;
$\mathsf{V}_{pm}(\mathsf{par}, \mathsf{crs}, \mathsf{x} = [\mathbf{a}]_2, \pi = (\pi_1, \ldots, \pi_n))$:
    1. $[a_n]_2 \stackrel{?}{=} (\sum_{i=1}^{n} [p_i(x)]_2) - (\sum_{i=1}^{n-1} [a_i]_2)$;
    2. For $i \in [1..n]$: $\mathsf{V}_{uv}(\mathsf{par}, \mathsf{crs}, [a_i]_2, \pi_i) \stackrel{?}{=} 1$;
```

Figure 4. Permutation matrix argument

divisible by $t(X)$. If the right-hand side is divisible by $t(X)$, then so must be the left-hand side which corresponds to $s(X)$.

Element α on the verifier side is used to batch those two equations together. Namely, if the verification equation in Fig. 3 is satisfied, then with an overwhelming probability also $[b]_1[1]_2 = [1]_1[a]_2$ and $([b]_1 + [p_0(x)]_1)([a]_2 + [p_0(x)]_2) - [1]_T = [\rho]_1[c]_2$ are satisfied. This allows us to reduce the number of expensive pairing operations.

Unfortunately, the above intuition for our argument is not perfectly correct. After carefully going through the proof in the generic group model, one finds that $[a]_2$ might contain an additive term $m_0[p_0(x)]_1$ where $m_0 \in \mathbb{Z}_p$ is not necessarily 0. We take a very simple approach here and just assume that the same-message argument respect to the same $[a]_2$ already accepted. This will essentially guarantee that $[a]_2$ is in the span of $\{[p_i(x)]_2\}_{i=1}^{n} \cup \{[\rho]_2\}$ and thus $m_0 = 0$. In Chapter 5 we provide a more elegant solution to this problem which merges the same-message argument and the unit vector argument.

Theorem 3. *The unit vector argument is perfectly complete and perfectly witness-indistinguishable. Assume that the same-message argument accepts, then the unit vector argument is knowledge sound in the GGM.*

3.4.1. Permutation Matrix Argument

From the unit vector argument, it is relatively easy to construct a permutation matrix argument. More precisely, we want to show that prover knows how to open n commitments to a permutation matrix, that is, to a $n \times n$ matrix where each row and each column is a unit vector.

Let us observe that matrix is a permutation matrix if and only if each of its columns is a unit vector and the sum of the columns is a vector of 1s. Following this idea, we will have a $\mathbf{p}(X)$-MP commitment $[a_i]_2$ for each column and prove that it is a unit vector. However, the last commitment $[a_n]_2$ we compute homomorphically,

$$[a_n]_2 \leftarrow \mathsf{Com}(\mathsf{ck}, \mathbf{1}_n; 0) - \big(\sum_{i=1}^{n-1}[a_i]_2\big) = \big(\sum_{i=1}^{n}[p_i(x)]_2\big) - \big(\sum_{i=1}^{n-1}[a_i]_2\big).$$

This will guarantee that $\sum_{i=1}^{n}[a_i]_2 = \mathsf{Com}(\mathsf{ck}, \mathbf{1}_n; 0)$, i.e., that columns sum to an all-one vector. For the sake of completeness, we also give the formal relation and the argument (see Fig. 4).

$$\mathscr{R}_{pm}^{(n)} = \left\{ \begin{array}{l} ([\mathbf{a}]_2, (\mathbf{A}, \mathbf{r})) \in \mathbb{G}_2^n \times (\mathbb{S}_n \times \mathbb{Z}_p^n) \mid \mathbf{a} = \mathbf{r} \cdot \rho + \mathbf{A}^\top \mathbf{p}(x) \wedge \\ \sum_{i=1}^{n} r_i = 0 \wedge \mathbf{A} \text{ is a permutation matrix} \end{array} \right\}.$$

Theorem 4. *Permutation matrix argument is perfectly complete, perfectly witness-indistinguishable, and knowledge sound in the GGM (under the same assumption as unit vector argument).*

3.5. Consistency Argument

The final sub-argument is the consistency argument which shows that n ciphertexts $[\mathbf{C}'_i]_1$ are in a linear span of ciphertexts $[\mathbf{C}]_1 \in \mathbb{G}_1^{n \times 2}$ and the public key pk. More formally we can phrase this as the following relation

$$\mathscr{R}_{con}^{(n)} = \left\{ \; \big([\mathbf{C}, \mathbf{C}']_1, (\mathbf{E}, \mathbf{t})\big) \in \mathbb{G}_1^{n \times 4} \times (\mathbb{Z}_p^{n \times n} \times \mathbb{Z}_p^n) \mid \mathbf{C}' = \mathbf{E} \cdot \mathbf{C} + \mathbf{t} \cdot \mathsf{pk} \; \right\}.$$

Observe that if we restrict the matrix $\mathbf{E}$ to be a permutation matrix then we actually obtain the shuffle relation in Eq. (3.1). We only prove culpable soundness of the consistency argument and do it respect to the following guilt relation

$$\mathscr{R}_{con\text{-}glt}^{(n)} = \left\{ \begin{array}{l} \big([\mathbf{C}, \mathbf{C}']_1, ([\hat{\mathbf{a}}]_2, \mathbf{E}, \hat{\mathbf{r}}, \mathsf{sk})\big) \in \mathbb{G}_1^{n \times 4} \times (\mathbb{G}_2^n \times \mathbb{Z}_p^{n \times n} \times \mathbb{Z}_p^n \times \mathbb{Z}_p) \mid \\ \mathsf{Dec}(\mathsf{sk}, \mathbf{C}') \neq \mathbf{E} \cdot \mathsf{Dec}(\mathsf{sk}, \mathbf{C}) \wedge \hat{\mathbf{a}} = \mathbf{E}^\top \cdot \mathbf{p}(x) + \hat{\mathbf{r}} \cdot \hat{\rho} \end{array} \right\}.$$

In other words, we will prove soundness against adversaries that know a vector of commitments $[\hat{\mathbf{a}}]_2$ and their openings $(\mathbf{E}, \mathbf{r})$ together with the decryption key. The proof is based on a variant of KerMDH assumption (in Chapter 5 we prove a similar result under a more minimalistic assumption) where the adversary gets some additional information as part of auxiliary input.

Let $\mathscr{D}_{con}$ be the CRS distribution for the whole shuffle argument in Fig. 6 (which includes output of $\mathsf{Cgen}_{con}(\mathsf{par})$) and let us divide the output to a matrix $[\mathbf{D}]_2 = [\begin{smallmatrix} \mathbf{q}(x) \\ \hat{\rho} \end{smallmatrix}]_2$ and the rest we will denote by aux.

```
$\mathsf{Cgen}_{con}(\mathsf{par})$: $x, \hat{\rho} \leftarrow_\$ \mathbb{Z}_p$; $\mathsf{crs} \leftarrow [\mathbf{q}(x), \hat{\rho}]_2$; Return $(\mathsf{crs}, \mathsf{td} = (x, \rho))$;
$\mathsf{P}_{con}(\mathsf{par}, \mathsf{crs}, \mathsf{x} = [\mathbf{C}, \mathbf{C}']_2, w = (\mathbf{E}, \mathbf{t}))$:
    1. $\hat{\mathbf{r}} \leftarrow_\$ \mathbb{Z}_p^n$; $[\hat{\mathbf{a}}]_2 \leftarrow \mathbf{E}^\top [\mathbf{q}(x)]_2 + \hat{\mathbf{r}}[\hat{\rho}]_2$;
    2. $r_t \leftarrow_\$ \mathbb{Z}_p$; $[s]_2 \leftarrow \mathbf{t}^\top [\mathbf{q}(x)]_2 + r_t[\hat{\rho}]_2$;
    3. $[\hat{\mathbf{c}}]_1 \leftarrow \hat{\mathbf{r}}^\top [\mathbf{C}]_1 + r_t \cdot \mathsf{pk}$;
    4. Return $\pi \leftarrow ([\hat{\mathbf{c}}]_1, [\hat{\mathbf{a}}, s]_2)$;
$\mathsf{V}_{con}(\mathsf{par}, \mathsf{crs}, \mathsf{x} = [\mathbf{C}, \mathbf{C}']_2, \pi = ([\hat{\mathbf{c}}]_1, [\hat{\mathbf{a}}, s]_2))$:
    1. $[\mathbf{C}']_1^\top [\mathbf{q}(x)]_2 - [\mathbf{C}]_1^\top [\hat{\mathbf{a}}]_2 \stackrel{?}{=} \mathsf{pk} \cdot [s]_2 - [\hat{\mathbf{c}}]_1 [\hat{\rho}]_2$;
```

Figure 5. Consistency argument

Definition 18 (KerMDH with aux)**.** *$\mathscr{D}_{con}$-KerMDH assumption with auxiliary input holds relative to* BPgen*, if for any PPT adversary* A*, $\varepsilon \approx_\lambda 0$, where*

$$\varepsilon := \Pr\left[\begin{array}{l} \mathsf{bp} \leftarrow \mathsf{BPgen}(1^\lambda), ([\boldsymbol{D}]_2, \mathsf{aux}) \leftarrow_\$ \mathscr{D}_{con}, [\boldsymbol{x}]_1 \leftarrow \mathsf{A}(\mathsf{bp}, [\boldsymbol{D}]_2, \mathsf{aux}) : \\ [\boldsymbol{x}]_1^\top [\boldsymbol{D}]_2 = [\boldsymbol{0}]_T \wedge [\boldsymbol{x}]_T \neq [\boldsymbol{0}]_T \end{array}\right].$$

We remind that so far we have fixed polynomials $\mathbf{p}(X)$ but have not fixed $\mathbf{q}(X)$. Fauzi et al. [FLSZ17b] prove that under a certain restriction on $\mathbf{q}(X)$, $\mathscr{D}_{con}$-KerMDH assumption with auxiliary input holds in GGM.[3]

Theorem 5. *Asssume that $\{q_i(X) \cdot p_j(X)\}_{i,j=1}^n$ is linearly independent. Then $\mathscr{D}_{con}$-KerMDH with auxiliary input holds in GGM.*

Let us fix $\mathbf{q}(X)$ for the sake of concreteness:

$$q_i(X) := X^{(i+1)(n+1)} \text{ for } i \in [1..n].$$

In particular, $\{q_i(X) \cdot p_j(X)\}_{i,j=1}^n$ is linearly independent.

Our version of the consistency argument, as presented in Fig. 5, is very similar to constructions by Groth and Lu [GL07] and Fauzi et al. [FL16, FLZ16]. To explain the intuition of the argument, let us make some simplifications to $\mathscr{R}_{con\text{-}glt}^{(n)}$ relation and to the consistency argument in Fig. 5: (i) commitment to $\mathbf{E}$ is without randomization, i.e., $[\hat{\mathbf{a}}]_2 = \mathbf{E}^\top [\mathbf{q}(x)]_2$, (ii) ciphertexts $[\mathbf{C}']_1$ are also not randomized, i.e., $[\mathbf{C}']_1 = \mathbf{E} \cdot [\mathbf{C}]_1$, and (iii) right hand side of the verification equation is $[\mathbf{0}]_T$, that is,

$$[\mathbf{C}']_1^\top [\mathbf{q}(x)]_2 - [\mathbf{C}]_1^\top [\hat{\mathbf{a}}]_2 \stackrel{?}{=} [\mathbf{0}]_T.$$

An adversary in the culpable soundness definition outputs (among other elements) $\mathbf{E}$ and a commitment $[\hat{\mathbf{a}}]_2 = \mathbf{E}^\top [\mathbf{q}(x)]_2$ such that $[\mathbf{C}']_1 \neq \mathbf{E}[\mathbf{C}]_1$. By substituting $[\hat{\mathbf{a}}]_2$ to the verification equation, we can rewrite it as $([\mathbf{C}']_1^\top - [\mathbf{C}]_1^\top \mathbf{E}^\top)[\mathbf{q}(x)]_2 = [\mathbf{0}]_T$. This, however, means that we have found a non-zero vector in the kernel

[3]Theorem in [FLSZ17b] is actually even more general than the one presented here.

of $[\mathbf{q}(x)]_2^\top$ and broken some form of KerMDH assumption. Real consistency argument follows the same idea but additionally takes randomization into account. That is why the argument in Fig. 5 also includes a commitment $[s]_2$ to message $\mathbf{t}$ and the ciphertext $[\hat{\mathbf{c}}]_1$.

Theorem 6. *Consistency argument is perfectly complete, perfectly zero-knowledge, and culpably sound respect to relation $\mathscr{R}^{(n)}_{con\text{-}glt}$ under $\mathscr{D}_{con}$-KerMDH assumption with auxiliary input.*

3.6. Shuffle Argument

Finally, in Fig. 6 we combine the same-message argument, the permutation matrix argument, and the consistency argument to obtain the shuffle argument $\mathcal{S}$. We start by committing to columns of permutation matrix $\mathbf{A}$ with $\mathbf{p}(X)$-MP commitment and $\mathbf{q}(X)$-MP commitment and obtain $[\mathbf{a}]_2$ and $[\hat{\mathbf{a}}]_2$. Randomness is picked such that $\sum_{i=1}^{n}[a_i]_2 = \mathsf{Com}(\mathsf{ck}, \mathbf{1}_n; 0)$ as we discussed in Section 3.4.1. We prove with the permutation matrix argument that prover knows an opening $(\mathbf{A}, \mathbf{r})$ of commitments $[\mathbf{a}]_2$ such that $\mathbf{A}$ is a permutation matrix and with the same-message argument we show that $[\hat{\mathbf{a}}]_2$ opens to the same $\mathbf{A}$ but possibly with a different randomness $\hat{\mathbf{r}}$. We give a consistency argument respect to the same $[\hat{\mathbf{a}}]_2$. Intuitively, the shuffle argument is sound since otherwise, we can construct an adversary that breaks culpable soundness of the consistency argument. In the reduction we can sample sk and pk ourselves, then run the adversary against the soundness of argument $\mathcal{S}$ and extract permutation matrix $\mathbf{A}$ and randomness $\mathbf{r}$ from the permutation argument and same-message argument respectively. Now we can return proof for the consistency argument together with the guilt witness $([\hat{\mathbf{a}}]_2, \mathbf{A}, \hat{\mathbf{r}}, \mathsf{sk})$. If $\mathsf{Dec}(\mathsf{sk}, \mathbf{C}') \neq \mathbf{E} \cdot \mathsf{Dec}(\mathsf{sk}, \mathbf{C})$, then we have broken culpable soundness and otherwise $[\mathbf{C}', \mathbf{C}]_1$ is a valid shuffle.

Theorem 7. *Shuffle argument $\mathcal{S}$ is perfectly complete and perfectly zero-knowledge. If commitments are binding and $\mathscr{D}_{con}$-KerMDH assumption with auxiliary input holds, then $\mathcal{S}$ is computationally sound in GGM.*

3.7. Batching and Implementation

It is possible to significantly increase the verification speed by using techniques for equation batching [BGR98]. It allows us to substitute most of the costly pairing operations with a lot less costly scalar multiplications in $\mathbb{G}_1$ or $\mathbb{G}_2$. For example, we need to verify that n same-message arguments $[d_1, \ldots, d_n]_2$ satisfy the equation $[1]_1[d_i]_2 = [K_1]_1[a_i]_2 + [K_2]_1[\hat{a}_i]_2$ for $i \in [1..n]$ which would take $3n$ pairing operations. Instead, the verifier can sample $z_1, \ldots, z_n \leftarrow_\$ \mathbb{Z}_p$ and check that

$$\sum_{i=1}^{n} z_i([1]_1[d_i]_2) = \sum_{i=1}^{n} z_i([K_1]_1[a_i]_2 + [K_2]_1[\hat{a}_i]_2).$$

```
Cgen(par):
  1. $(\mathsf{crs}_{pm}, (x, \rho)) \leftarrow \mathsf{Cgen}_{pm}(\mathsf{par})$; $\hat{\rho} \leftarrow_{\$} \mathbb{Z}_p$; $\mathsf{crs}_{con} \leftarrow [\mathbf{q}(x), \hat{\rho}]_2$;
  2. $[\mathbf{M}]_2 = \begin{bmatrix} \mathbf{p}(x) & \rho & 0 \\ \mathbf{q}(x) & 0 & \hat{\rho} \end{bmatrix}_2$; $(\mathsf{crs}_{sm}, \mathsf{td}_{sm}) \leftarrow \mathsf{Cgen}_{sm}(\mathsf{par}, [\mathbf{M}]_2)$;
  3. Return $(\mathsf{crs} = (\mathsf{crs}_{pm}, \mathsf{crs}_{sm}, \mathsf{crs}_{con}), \mathsf{td} = (x, \hat{\rho}))$;
P(par, crs, x = [C, C']_2, w = (A, t)):
  1. $\mathbf{r}, \hat{\mathbf{r}} \leftarrow_{\$} \mathbb{Z}_p^n$ such that $\sum_{i=1}^{n} r_i = 0 = \sum_{i=1}^{n} \hat{r}_i$;
  2. $[\mathbf{a}]_2 \leftarrow \mathbf{A}^\top \cdot [\mathbf{p}(x)]_2 + \mathbf{r}[\rho]_2$; $[\hat{\mathbf{a}}]_2 \leftarrow \mathbf{A}^\top \cdot [\mathbf{q}(x)]_2 + \hat{\mathbf{r}}[\hat{\rho}]_2$;
  3. $\pi_{pm} \leftarrow \mathsf{P}_{pm}(\mathsf{par}, \mathsf{crs}_{uv}, [\mathbf{a}]_2, (\sigma, \mathbf{r}))$;
  4. $r_t \leftarrow_{\$} \mathbb{Z}_p$; $[s]_2 \leftarrow \mathbf{t}^\top [\mathbf{q}(x)]_2 + r_t [\hat{\rho}]_2$; $[\hat{\mathbf{c}}]_1 \leftarrow \hat{\mathbf{r}}^\top [\mathbf{C}]_1 + r_t \cdot \mathsf{pk}$;
  5. For $i \in [1..n]$: $\pi_{sm:i} \leftarrow \mathsf{P}_{sm}(\mathsf{par}, \mathsf{crs}_{sm}, [a_i, \hat{a}_i]_2, (\mathbf{A}_i, r_i, \hat{r}_i))$;
  6. $\pi_{con} = ([\hat{\mathbf{c}}]_1, [\hat{\mathbf{a}}, s]_2)$; $\pi_{sm} \leftarrow (\pi_{sm:i})_{i=1}^{n}$;
  7. Return $\pi \leftarrow ([\mathbf{a}]_2, \pi_{pm}, \pi_{con}, \pi_{sm})$;
V(par, crs, x = [C, C']_2, π = ([a]_2, π_pm, π_con, π_sm)):
  1. $\mathsf{V}_{pm}(\mathsf{par}, \mathsf{crs}_{pm}, [\mathbf{a}]_2, \pi_{pm}) \overset{?}{=} 1$;
  2. $\mathsf{V}_{con}(\mathsf{par}, \mathsf{crs}_{con}, [\mathbf{C}, \mathbf{C}']_2, \pi_{con}) \overset{?}{=} 1$;
  3. For $i \in [1..n]$ : $\mathsf{V}_{sm}(\mathsf{par}, \mathsf{crs}, [a_i, \hat{a}_i]_2, \pi_{sm:i}) \overset{?}{=} 1$;
```

Figure 6. Shuffle argument $\mathcal{S}$

This by itself would give no efficiency gain, but we can use bilinearity properties to write it as

$$[1]_1 \cdot (\sum_{i=1}^{n} z_i [d_i]_2) = [K_1]_1 \cdot (\sum_{i=1}^{n} z_i [a_i]_2) + [K_2]_1 \cdot (\sum_{i=1}^{n} z_i [\hat{a}_i]_2).$$

This equation only needs 3 pairings and $3n$ scalar multiplications. Depending on the pairing construction, scalar multiplication in $\mathbb{G}_2$ can be several times faster than a pairing operation (see [FLSZ17b] for concrete numbers). One can further improve exponentiation time by sampling z_i from a smaller set, say $\{0,1\}^{40} \subset \mathbb{Z}_p$. It is possible to show that the probability that one of the original equations would not accept but the batched equation would accept is bounded by $1/2^{40}$.

Furthermore, we implement the whole shuffle argument[4] in C++ library Libsnark together with batching and several other optimizations. We tested our argument on a small laptop computer (Intel Core i5-CPU with 4 threads, 8 GB RAM, 64bit Linux Ubuntu 16.10) and even on such modest hardware achieved quite

[4]Implementation is available at `https://bitbucket.org/JannoSiim/hat_shuffle_implementation/`

Table 3. Running times for $n = 10\,000, 100\,000, 300\,000$ ciphertexts

	10,000	100,000	300,000
CRS generation	2.6s	20.5s	55.0s
Prover	9.1s	1m 24s	4m 2.4s
Prover (online)	0.5s	4.1s	13.5s
Verifier	8.3s	1m 22.0	4m 49.2s
Verifier (online)	5.0s	49.9s	2m 55.5s

good running times. In Table 3 we have running times for CRS generation, proof generation, and verification time for $n = 10\,000, 100\,000, 300\,000$ ciphertexts.

Some of the computation can be done offline before the ciphertexts are even known (e.g., permutation matrix argument and the same-message argument). For the i-voting scenario, this computation can be done weeks before the elections. Other computations are online computation, that is, ciphertexts need to be available (shuffling and the consistency argument). We see from the table that online computation, especially for the prover, is very fast for our argument.

4. UC-SECURE CRS GENERATION FOR NIZK

4.1. Motivation

Most of the current NIZK arguments are secure either in the RO model or in the CRS model (sometimes even both, e.g. [BG18]). By folklore reasoning, uniformly random CRS can be obtained from some physical source of randomness (stock market fluctuations, sunspot data, etc.) [1]. The same does not seem to be possible with a more structured distribution like $[x, x^2]$ over some elliptic curve group. This raises a major issue with CRS based NIZK arguments: *How should CRS be obtained?* One option is to include a trusted third party that generates the CRS and does not leak the trapdoor or any other side-information, but this is a very strong trust assumption that might not be warranted in practice.

We can reduce trust by distributing CRS generation among multiple independent parties. This is can be done with a secure multi-party computation protocol that outputs a CRS but the secret information (trapdoor) is shared among all parties. Such an approach was considered by Ben Sasson et al. [BCG$^+$14]. According to their analysis, if we want to involve many parties (many of them possibly malicious), then the current generic secure multi-party computation protocols are inefficient for CRS generation for SNARKs or our shuffle argument. Thus they proposed their own distributed CRS generation protocol which works for a large class of pairing-based NIZK arguments. It shows good efficiency and requires only 1 out of N parties to be honest. Unfortunately, it also assumes the RO model and that the CRS is symmetric, i.e., discrete logarithms of $\mathbb{G}_1$ elements are the same as discrete logarithms of $\mathbb{G}_2$ elements. Thus, it does not fit with the structure and the security model of our shuffle argument $\mathcal{S}$.

Motivated by this, we propose in [ABL$^+$19b] an improvement to Ben-Sasson et al.'s protocol and remove both of the restrictions.

Removing the RO model : We observe that [BCG$^+$14] requires the RO model since they use non-interactive sigma protocols to show that each party knows a share of the trapdoor. In our construction, we will substitute sigma protocols with a UC-secure DL-extractable commitment scheme [ABL$^+$19a]. We have each party commit to their trapdoor share ts_i and once everyone has committed they will open the commitment to $[\mathsf{ts}_i]_1$, $[\mathsf{ts}_i]_2$. On the one hand, it guarantees that trapdoor shares are independently chosen and on the other hand it is possible to extract ts_i from the commitment just as in [BCG$^+$14]. It turns out that by using a UC-secure DL-extractable commitment we get UC-security for the whole CRS generation protocol. This works well with our objective since it means that elements leaked in the CRS generation protocol cannot affect soundness or zero-knowledge properties.

[1]Even with uniform CRS, the situation is not so simple. Natural distributions are typically non-uniform and thus cannot directly be used as a CRS.

Asymmetric CRS : It turns out to be possible to change the original protocol such that there is only partial symmetry between $\mathbb{G}_1$ and $\mathbb{G}_2$ elements of the CRS. Namely we require that for each element $\alpha \in \mathsf{td}$, CRS contains $[\alpha]_1$ and $[\alpha]_2$. Besides that, CRS does not need to be symmetrical. This change also means that the proof strategy must be slightly different compared to the original protocol. Other restrictions on the CRS in [BCG$^+$14] will stay mostly the same for our protocol. We will cover them in detail in Section 4.3.1.

Overview. We start by explaining the main idea of the protocol with an example. Suppose that parties $\mathsf{P}_1, \dots, \mathsf{P}_N$ want to generate a CRS $[\alpha, \alpha^2]_1$ where $\alpha \leftarrow_\$ \mathbb{Z}_p^*$. We have each party P_i generate a share $\alpha_i \leftarrow_\$ \mathbb{Z}_p$ and we implicitly define $\alpha := \prod_{i=1}^{N} \alpha_i$. We would like each party to reveal $[\alpha_i]_2$ but such that shares are guaranteed to be picked independently and thus α is uniformly random in $\mathbb{Z}_p^*$. This problem is easily solved by starting the protocol with a round where parties broadcast DL-extractable commitments of $[\alpha_i]_2$ and open it in the next round. Once this is done, parties can start computing the actual CRS. This will proceed in a round-robin fashion: P_1 broadcasts $\alpha_1[1]_1$, P_2 broadcasts $\alpha_2[\alpha_1]_1$, and so on until P_N broadcasts $\alpha_N[\alpha_1 \cdot \ldots \cdot \alpha_{N-1}]_1 = [\alpha]_1$. Importantly, each computation can be verified by each party. Say if P_{i-1} outputted $[a_{i-1}]_1 = [\alpha_1 \cdot \cdots \ldots \alpha_{i-1}]_1$ and we want to verify output $[a_i]_1$ of party P_i, then it can be done with a pairing equation $[a_i]_1[1]_2 = [a_{i-1}]_1[\alpha_i]_2$. To get $[\alpha^2]_1$, we add one more round-robin phase: P_1 broadcasts $\alpha_1[\alpha]_1$, P_2 broadcasts $\alpha_2[\alpha\alpha_1]_1$, until P_N broadcasts $\alpha_N[\alpha\alpha_1 \ldots \alpha_{N-1}]_1 = [\alpha^2]_1$. Verification can be done as before.

One caveat here is that if say parties $\mathsf{P}_1, \dots, \mathsf{P}_{N-1}$ are corrupted, then they can compute also $[\alpha]_2$ just from $\alpha_1, \dots, \alpha_{N-1}$ and $[\alpha_N]_2$. In some NIZK arguments, this might affect soundness or zero-knowledge properties. Therefore, we will explicitly require that for each trapdoor element x, CRS must contain both $[x]_1$ and $[x]_2$. Thus in our example, CRS should actually be $[\alpha, \alpha^2]_1$, $[\alpha]_2$. Full protocol in Section 4.3 follows the same ideas, but we additionally allow addition (and division) which makes the protocol more involved.

4.2. Ideal Functionality

First, we give a UC ideal functionality for pairing-based CRS generation protocols. We start by formalizing a pairing-based CRS. Let $C_1 : \mathbb{Z}_p^t \to \mathbb{Z}_p^{h_1}$ and $C_2 : \mathbb{Z}_p^t \to \mathbb{Z}_p^{h_2}$ be two arithmetic circuits such that $\mathsf{crs} = \big([C_1(\mathsf{td})]_1, [C_2(\mathsf{td})]_2\big)$ for $\mathsf{td} \leftarrow_\$ \mathscr{D}$ where $\mathscr{D}$ is some distribution over $\mathbb{Z}_p^t$. Protocol will involve parties $\mathsf{P}_1, \dots, \mathsf{P}_N$. Each party P_i will have a share ts_i of trapdoor td. Honest parties will have their ts_i sampled from $\mathscr{D}$ by the ideal functionality whereas malicious parties can submit their own ts_i as long as $\mathsf{ts}_i \in \mathscr{D}$. Ideal functionality combines shares $\mathsf{ts}_1, \dots, \mathsf{ts}_N$ to trapdoor td using some combining function $c : (\mathbb{Z}_p^t)^N \to \mathbb{Z}_p^t$. Once

Let parties be $\mathsf{P}_1, \ldots, \mathsf{P}_N$ and let $C_1, C_2, \mathscr{D}$ define the CRS.

Share collection from honest P_i: On input $(\mathsf{collect}, \mathsf{sid}, \mathsf{P}_i)$ from honest P_i check that no message with $\mathsf{collect}$ and sid has been stored. If so, then sample $\mathsf{ts}_i \leftarrow_{\$} \mathscr{D}$, store $(\mathsf{collect}, \mathsf{sid}, \mathsf{P}_i, \mathsf{ts}_i)$, and send $(\mathsf{collect}, \mathsf{sid}, \mathsf{P}_i)$ to Sim.

Share collection from malicious P_i: On input $(\mathsf{collect}, \mathsf{sid}, \mathsf{P}_i, \mathsf{ts}_i)$ from malicious P_i, check that $\mathsf{ts}_i \in \mathscr{D}$ and that no message with $\mathsf{collect}$ and sid has been stored. If so, then store $(\mathsf{collect}, \mathsf{sid}, \mathsf{P}_i, \mathsf{ts}_i)$.

CRS generation: Once ts_i is stored for each $i \in [1..N]$:

1. Compute $\mathsf{td} \leftarrow c(\mathsf{ts}_1, \ldots, \mathsf{ts}_N)$.
2. Set $\mathsf{crs} \leftarrow \left([C_1(\mathsf{td})]_1, [C_2(\mathsf{td})]_2\right)$ and send $(\mathsf{output}, \mathsf{sid}, \mathsf{crs})$ to Sim.
3. If Sim returns $(\mathsf{output}, \mathsf{sid}, \mathsf{ok})$ then send $(\mathsf{output}, \mathsf{sid}, \mathsf{crs})$ to all P_i for $i \in [1..N]$.

Figure 7. Ideal functionality $\mathsf{F}_{\mathsf{dcrs}}$ for distributed CRS generation

ideal functionality has trapdoor shares from all parties, it will compute the CRS from td and return it to all parties. Full details of the ideal functionality $\mathsf{F}_{\mathsf{dcrs}}$ can be found in Fig. 7.

This ideal functionality does what we expect if the adversary cannot distinguish between a CRS generated by an honest party and a CRS generated by $\mathsf{F}_{\mathsf{dcrs}}$. Firstly, we cannot guarantee it for any non-trivial CRS if all parties are corrupted. Thus, we must allow the adversary to corrupt at most $N-1$ parties. Secondly, if $\mathsf{td} \leftarrow_{\$} \mathscr{D}$ and $\mathsf{td} \leftarrow c(\mathsf{ts}_1, \ldots, \mathsf{ts}_N)$ are indistinguishable then so is the CRS. Hence, we should fix the combining function c such that the adversary would not be able to distinguish between $\mathsf{td} \leftarrow_{\$} \mathscr{D}$ and $\mathsf{td} \leftarrow c(\mathsf{ts}_1, \ldots, \mathsf{ts}_N)$ when there is at least one honest party. There are two natural settings for this.

- If $\mathscr{D}$ is uniform distribution over $\mathbb{Z}_p^t$, then c could be point-wise addition.
- If $\mathscr{D}$ is $(\mathbb{Z}_p^*)^t$, then c could be point-wise multiplication.

Our protocol will use the latter option.

4.3. Protocol

4.3.1. Class of Circuits

We describe the class of CRSs that our protocol can securely generate. Our presentation here significantly differs from [ABL+19b] and will be more similar to [BGM17] which is hopefully more digestible.

Firstly, we assume that trapdoors td are distributed uniformly over $(\mathbb{Z}_p^*)^t$ and later our combining function c will be pointwise multiplication. Thus, if we have shares $\mathsf{ts}_1, \ldots, \mathsf{ts}_N \in (\mathbb{Z}_p^*)^t$ and at least one of them is sampled uniformly and independently, then $c(\mathsf{ts}_1, \ldots, \mathsf{ts}_N)$ is also distributed uniformly over $(\mathbb{Z}_p^*)^t$.

Let parties be $\mathsf{P}_1, \ldots, \mathsf{P}_N$.

- On input $(\mathsf{broadcast}, \mathsf{sid}, \mathsf{P}_i, x)$ from party P_i, send $(\mathsf{broadcast}, \mathsf{sid}, \mathsf{P}_i, x)$ to P_i for $i \in [1..N] \setminus \{i\}$ and Sim.

Figure 8. Ideal functionality F_{bc} for authenticated broadcast

Secondly, we need to describe circuits $C_k : \mathbb{Z}_p^t \to \mathbb{Z}_p^{h_k}$ for $k \in \{1,2\}$. The circuit will contain (i) multiplication-division gates that have three input wires with values x, y, z and an output wire with value $(x \cdot y)/z$, and (ii) addition gates with some constant number e of input wires with values $x_1, \ldots, x_e$ and with an output wire with value $\sum_{i=1}^{e} a_i x_i$ where a_i for $i \in [1..e]$ are some constants. All gates in C_k can be divided into disjoint layers $M_1, A_2, \ldots, M_{d-1}, A_d$ where M_i contains only multiplication-division gates and A_i contains only addition gates. Moreover, input wires of a gate in i-th layer are from a gate at the same or from a lower layer or are directly circuit inputs. For efficiency reasons, it is always best to partition gates such that the number of layers is minimum. Additionally, we will have the following restrictions for gates:

1. The output of each gate will also be the output of the whole circuit. This will allow us to publicly verify the computation of each gate.
2. For each input td_i of the circuit we have one trivial multiplication-division gate (as part of M_1) which outputs $(1 \cdot \mathsf{td}_i)/1 = \mathsf{td}_i$. This means that CRS will contain $([\mathsf{td}_i]_1, [\mathsf{td}_i]_2)$ which will be essential to simulate the protocol in the UC-security proof.
3. Multiplication-division gates must always have left input wire (wire with x) to be the output of another gate (or constant) and middle and right input wires (respectively wires with y and z) as one of the inputs td_i of the circuit (or constant). In the protocol, this will allow us to verify the computation of the gate with simple pairing-based equations.

4.3.2. Construction

We assume to have access to an authenticated broadcast [CLOS02] defined in Fig. 8 and DL-extractable commitment in Fig. 1. Our setting is very similar to multi-party protocol SPDZ [DPSZ12]. Namely, we allow $N-1$ out N parties to be malicious which means that successful termination cannot be guaranteed. If cheating is detected by an honest party, he will simply abort the protocol. We assume in the following that C_1 and C_2 follow description from Section 4.3.1.

Share collection phase. Honest party P_i receives an input $(\mathsf{collect}, \mathsf{sid}, \mathsf{P}_i)$ from the environment which indicates that the share collection phase has started. Party P_i samples trapdoor shares $\mathsf{ts}_i = (\mathsf{ts}_{i1}, \ldots, \mathsf{ts}_{it})$ uniformly from $(\mathbb{Z}_p^*)^t$. It proceeds by sending a DL-extractable commitment of ts_i to each party and then waits until he has received commitments from all other parties. Once this has happened, P_i sends opening (which includes $[\mathsf{ts}_i]_1$) to each party and additionally calls $(\mathsf{broadcast}, \mathsf{sid}, \mathsf{P}_i, [\mathsf{ts}_i]_1, [\mathsf{ts}_i]_2)$. This will guarantee that (i) some malicious party P_j cannot send a commitment of ts_j to one party and a commitment of different message ts_j^* to another party, (ii) $[\mathsf{ts}_i]_2$ will also be publicly known by all parties. That means P_i will check that (i) $[\mathsf{ts}_j]_1$ received as part of an opening and $[\mathsf{ts}_j^*]_1$ received from the broadcast are equal, (ii) shares in both groups are equal, i.e., $[\mathsf{ts}_j]_1[1]_2 = [1]_1[\mathsf{ts}_j]_2$. He will abort if either of the checks fails. Party will wait until all messages are received from other parties and only then will proceed to the CRS generation phase.

CRS generation phase. At this point, each party knows $[\mathsf{ts}_j]_1, [\mathsf{ts}_j]_2$ for $j \in [1..N]$ and the rest of the computation will be a deterministic evaluation of circuits C_1 and C_2. Parties will evaluate circuits layer-by-layer and C_1 and C_2 can be processed in parallel. Multiplication-division layers are evaluated in a round-robin fashion, that is, parties will include their shares one by one to the total result. Therefore, evaluation of one layer M_i takes exactly N rounds to complete. Addition layers can be computed locally as we will see.

Let us describe the evaluation of one multiplication-division layer M of circuit C_1 (C_2 layer would be similar).

Evaluation starts with party P_1 and he will evaluate gates of M in topological order, i.e., if gate G_t depends on gate G_s then G_s is evaluated before G_t. Suppose that P_1 is evaluating multiplication-division gate G

For the left wire there are three options:

- If the left wire of G is some constant c, then we set $[x]_1 \leftarrow [c]_1$.
- If the left wire of G is from a lower layer, then this gate has been already evaluated and we can set $[x]_1$ to this value. Actually, if we minimize the number of layers, then the left wire can only come from the addition gate (or is a constant). Otherwise, G itself should be from a lower layer.
- If the left wire is from some multiplication-division gate G' of the same layer M. Then due to evaluation in topological order, P_1 has already computed output for G' and we set $[x]_1$ to be this value.

The middle wire value and the right wire value are treated as follows.

- If the middle wire (respectively right wire) is a constant c, then we set $y \leftarrow c$ (respectively $z \leftarrow c$).
- If the middle wire (respectively the right wire) corresponds to trapdoor input td_j, then we set $y \leftarrow \mathsf{ts}_{1,j}$ (respectively $z \leftarrow \mathsf{ts}_{1,j}$).

Now P_1 will broadcast $(y/z) \cdot [x]_1 = [(xy)/z]_1$ as his local evaluation of gate G. Importantly all parties will know $[x]_1$, $[y]_2$, and $[z]_2$ which means that they can verify this output with the equation $[xy/z]_1[z]_2 = [x]_1[y]_2$. Party will abort if verification does not pass.

Subsequent parties will proceed slightly differently. Suppose that P_{i-1} has finished his computation and it is now P_i's turn. Evaluation of gates is still done in the same order, but with a small modification. Namely, P_i should include his shares to the computation of the previous party.

We do it with the following algorithm. To evaluate gate G we first set $[x]_1$ to be P_{i-1}'s output for gate G and we set a temporary value $G' \leftarrow G$.

While $G' \in M$ we perform the following:

1. If the middle wire of G' has a constant c, then we set $y \leftarrow 1$ (P_1 already included constant c) and if it corresponds to td_j, then we set $y \leftarrow \mathsf{ts}_{i,j}$.
2. If the right wire of G' has a constant c, then we set $z \leftarrow 1$ (P_1 already included constant c) and if it corresponds to td_j, then we set $z \leftarrow \mathsf{ts}_{i,j}$.
3. P_i will broadcast element $[x']_1 \leftarrow (y/z)[x]_1$ and each party will verify it with the equation $[x']_1[z]_2 = [x]_1[y]_2$.
4. We set $[x]_1 \leftarrow [x']_1$ and new G' to be the left gate of the current G' or $\perp$ if there is no left gate.

We consider last broadcasted value to be P_i's evaluation of gate G. All the intermediate values are used just to check that this value is computed correctly. When the final party P_N has finished evaluating G, then P_N's last output is the actual element of the CRS. In particular, it will include shares of all the parties in the correct way.

Evaluating gates on some addition layer A is much easier and can be done locally by each party. Evaluation proceeds again in topological order. For each input wire of an addition gate, we already know the final value $[x_j]_1$:

1. If a wire comes from another addition gate, then we just use the locally computed evaluation of that.
2. If it comes from a multiplication-division gate then we will use P_N's evaluation of that gate.
3. If the wire is from the input of the circuit for some trapdoor td_i, then we use P_N's evaluation of the trivial gate $[1 \cdot \mathsf{td}_i/1]_1$.

Therefore each party can locally compute $\sum_{j=1}^{e} a_j [x_j]_1$.

In [ABL$^+$19b], we prove the following theorem.

Theorem 8. *The protocol above UC-realizes* $\mathsf{F}_{\mathsf{dcrs}}$ *functionality in* $(\mathsf{F}_{\textit{dl-com}}, \mathsf{F}_{\mathsf{bc}})$*-hybrid model.*

We remind that instantiation of $\mathsf{F}_{\textit{dl-com}}$ itself might require a trust assumption and this depends on the concrete construction of the DL-extractable commitment. We refer the reader to Section 2.6 for some possible instantiations, e.g., in the RPK model or in the uniform random string model.

5. TRANSPARENT SHUFFLE ARGUMENT

5.1. Overview

This section follows the results in [AFK$^+$20]. Our goal is to reduce trust in the shuffle argument $\mathcal{S}$ from Chapter 3 as much as possible. We achieve it in two ways. Firstly, we apply the distributed CRS generation protocol from the previous chapter. This guarantees that soundness and zero-knowledge will hold if at least one out of $N-1$ parties in CRS generation protocol are honest. Unfortunately, CRS of $\mathcal{S}$ does not fit into restrictions laid out in section Section 4.3.1. This requires us to make several non-trivial modifications to $\mathcal{S}$.

Secondly, we make our shuffle subversion zero-knowledge by following ideas of [BFS16,ABLZ17,Fuc18]. In particular, we define a CRS verification algorithm CV such that if CRS passes this verification, then zero-knowledge will hold even if CRS came from an untrusted party or equivalently if all N parties in the CRS generation protocol were dishonest.

We also make some other modifications along the way. We will merge the same-message argument with the unit vector argument since their security does not hold in isolation as we discussed in Chapter 3. We also skip the consistency argument since it has only culpable soundness and we directly construct shuffle argument from the permutation argument. We also prove in the paper that the new unit vector argument (merge of unit vector argument and same-message argument) has knowledge soundness in the algebraic group model (AGM) under the SPDL assumption. This is an improvement over GGM proof in [FLSZ17b]. Shuffle argument's soundness holds if permutation argument is knowledge sound and a novel kernel-type assumption holds that we call GapKerMDH assumption. Although this assumption is new, we believe it to be more natural than KerMDH assumption with auxiliary input in Definition 18. We prove subversion zero-knowledge under the knowledge assumption BDH-KE, similar to Abdolmaleki et al. [ABLZ17].

5.2. Modifications in CRS

CRS of $\mathcal{S}$ needs several modifications to satisfy structural requirements for our CRS generation protocol. We list the modifications below.

1. First, we observe that setting $\hat{\rho} = 1$ is secure, i.e., we can actually remove this trapdoor element.
2. We add $[x]_1$, $[x]_2$, and $[\mathbf{K}]_2 = [K_1, K_2]_2^\top$ to the CRS so that each trapdoor is a discrete logarithm of a $\mathbb{G}_1$ element and a $\mathbb{G}_2$ element.
3. Since we need to compute $[p_i(x)]_1$, $[p_i(x)]_2$, and $[((p_i(x)+p_0(x))^2-1)/\rho]_2$ for $i \in [1..n]$, we include to the CRS $\{[x^i]_1\}_{i=1}^n$ and $\{[x^i]_2\}_{i=1}^{2n}$. This will allow us to compute $[p_i(x)]_1$, $[p_i(x)]_2$, and $[(p_i(x)+p_0(x))^2-1]_2$ with addi-

tion gates and finally $[((p_i(x)+p_0(x))^2-1)/\rho]_2$ with multiplication-division gates.

4. Polynomials $q_i(X)$ in the original argument have the degree $(i+1)(n+1)$. This would imply that we need to include $\{[x^i]_2\}_{i=1}^{(n+1)^2}$ to the CRS. We will avoid this quadratic blowup by introducing a new random point θ and by redefining the polynomials $q_i(X)$. Namely, we will have $q_i^+(X_\theta) := X_\theta^{2i}$ for $i \in [1..n]$ and thus will use $\mathbf{q}^+(X_\theta)$-MP commitment scheme.[1] Additionally, we will need to include $[\theta]_1$ and $\{[\theta^i]_2\}_{i=1}^{2n}$ to the CRS.
5. Finally, in order to compute $[K_1 p_i(x) + K_2 \hat{q}_i(x)]_2$, we would need to reveal $[K_1 p_i(x)]_2$ and $[K_2 \hat{q}_i(x)]_2$. This, however, makes the unit vector argument insecure. Instead, we will modify the unit vector argument such that it uses $K_1[K_1 p_i(x) + K_2\hat{q}_i(x)]_2 = [K_1^2 p_i(x) + K_1K_2\hat{q}_i(x)]_2$, $[K_1\rho]_2$, and $[K_1K_2]_2$. In such case, we can reveal $[K_1 p_i(x)]_2$, $[K_2\hat{q}_i(x)]_2$ for $i \in [1..n]$ and also $[K_1\rho]_2$ without affecting the security of the unit vector argument.

It is now straightforward to verify that the CRS satisfies the necessary circuit structure. We refer the reader to [AFK+20, Fig. 7] for an explicit description of the circuit.

5.3. Unit Vector Argument

As mentioned before, the unit vector argument in $\mathcal{S}^+$ is a merge of the unit vector argument and the same-message argument in $\mathcal{S}$ and additionally instead of $[\mathbf{K}]_2 = [K_1, K_2]_2$ we will use $[\mathbf{K}^+]_2 = [K_1^2, K_1K_2]_2$. a detailed description is given in Fig. 10.

New argument is for the relation

$$\mathscr{R}_{uv+}^{(n)} = \{\ ([\hat{a}]_2, (I, \hat{r})) \in \mathbb{G}_2 \times ([1..n] \times \mathbb{Z}_p) \mid \hat{a} = q_I^+(x) + \hat{r}\ \},$$

and in [AFK+20] we prove the following theorem.

Theorem 9. *The new unit vector argument is perfectly complete and perfectly zero-knowledge. If $(3n-1)$-SPDL assumption holds, then it has computational knowledge soundness.*

5.4. Permutation Argument

Construction of the permutation argument (see Fig. 10) is essentially identical to the construction of the permuation matrix argument in $\mathcal{S}$. However, for a slightly different relation

$$\mathscr{R}_{pm+}^{(n)} = \left\{ \begin{array}{l} ([\hat{\mathbf{a}}]_2, (\mathbf{A}, \hat{\mathbf{r}})) \in \mathbb{G}_2^n \times (\mathbb{Z}_p^{n\times n} \times \mathbb{Z}_p^n) \mid \hat{\mathbf{a}} = \mathbf{A}^\top \mathbf{q}^+(x) + \hat{\mathbf{r}} \wedge \\ \sum_{i=1}^n \hat{r}_i = 0 \wedge \mathbf{A} \text{ is a permutation matrix} \end{array} \right\}.$$

[1]The more obvious definition of $q_i^+(X_\theta) = X_\theta^i$ turns out to be insecure as we explain in [AFK+20].

```
$\mathsf{Cgen}^{+}_{uv}(\mathsf{par})$:
  1. $x, \rho, \theta \leftarrow_{\$} \mathbb{Z}_p$; $\mathbf{K} = (K_1, K_2)^\top \leftarrow_{\$} \mathbb{Z}_p^2$;
  2. $\mathbf{u} \leftarrow \{p_i(x)\}_{i=0}^{n} \cup \{\rho\}$; $\mathbf{v} \leftarrow \{\big((p_i(x) + p_0(x))^2 - 1\big)/\rho\}_{i=1}^{n}$;
     $\mathbf{w} \leftarrow \{q_i^{+}(\theta)\}_{i=1}^{n}$;
  3. $\mathbf{M} = \begin{pmatrix} \mathbf{p}(x) & \rho & 0 \\ \mathbf{q}^{+}(\theta) & 0 & 1 \end{pmatrix}$; $\mathbf{K}^{+} \leftarrow (K_1^2, K_1 K_2)^\top$; $\mathbf{P} \leftarrow \mathbf{M}^\top \cdot \mathbf{K}^{+} \in \mathbb{Z}_p^{n \times 1}$;
  4. Return $(\mathsf{crs} = ([\mathbf{u}, \mathbf{K}^{+}]_1, [\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{P}]_2), \mathsf{td} = (x, \rho, \theta, \mathbf{K}))$;
$\mathsf{P}^{+}_{uv}(\mathsf{par}, \mathsf{crs}, \mathsf{x} = [\hat{a}]_2, w = (I, \hat{r}))$:
  1. $r \leftarrow_{\$} \mathbb{Z}_p$; $[b]_1 \leftarrow [p_I(x)]_1 + r[\rho]_1$; $[a]_2 \leftarrow [p_I(x)]_2 + r[\rho]_2$;
  2. $[c]_2 \leftarrow 2r[a]_2 + 2r[p_0]_2 - r^2[\rho]_2 + [((p_I(x) + p_0(x))^2 - 1)/\rho]_2$;
  3. $\pi_{sm} \leftarrow [P]_2^\top \cdot (\mathbf{e}_I^\top, r, \hat{r})^\top$; //$\mathbf{e}_I$ is the $I$-th unit vector
  4. Return $\pi = ([b]_1, [a, c]_2, \pi_{sm})$;

$\mathsf{V}^{+}_{uv}(\mathsf{par}, \mathsf{crs}, \mathsf{x} = [\hat{a}]_2, \pi = ([b]_1, [a, c]_2, \pi_{sm}))$:
  1. $\alpha \leftarrow_{\$} \mathbb{Z}_p$;
  2. $([b]_1 + [\alpha]_1 + [p_0(x)]_1)([a]_2 - [\alpha]_2 + [p_0(x)]_2) \stackrel{?}{=} [\rho]_1 [c]_2 + [1 - \alpha^2]_T$;
  3. $[\mathbf{K}^{+}]_1^\top [a, \hat{a}]_2^\top \stackrel{?}{=} [1]_1 \cdot \pi_{sm}$;
```

Figure 9. New unit vector argument

```
$\mathsf{Cgen}^{+}_{pm}(\mathsf{par})$: Return $\mathsf{Cgen}^{+}_{uv}(\mathsf{par})$;
$\mathsf{P}^{+}_{pm}(\mathsf{par}, \mathsf{crs}, \mathsf{x} = [\hat{\mathbf{a}}]_2, w = (\mathbf{A}, \hat{\mathbf{r}}))$:
  1. For $i \in [1..n]$: $\pi_i \leftarrow \mathsf{P}^{+}_{uv}(\mathsf{par}, \mathsf{crs}, [\hat{a}_i]_2, (\mathbf{A}_i^\top, r_i))$;
  2. Return $\pi = (\pi_1, \dots, \pi_n)$;
$\mathsf{V}^{+}_{pm}(\mathsf{par}, \mathsf{crs}, \mathsf{x} = [\hat{\mathbf{a}}]_2, \pi = (\pi_1, \dots, \pi_n))$:
  1. $[\hat{a}_n]_2 \stackrel{?}{=} \big(\sum_{i=1}^{n} [q_i^{+}(x)]_2\big) - \big(\sum_{i=1}^{n-1} [\hat{a}_i]_2\big)$;
  2. For $i \in [1..n]$: $\mathsf{V}^{+}_{uv}(\mathsf{par}, \mathsf{crs}, [\hat{a}_i]_2, \pi_i) \stackrel{?}{=} 1$;
```

Figure 10. New permutation argument

We will also prove a slightly stronger result compared to permutation matrix argument of $\mathcal{S}$.

Theorem 10. *Permutation argument is perfectly complete and perfectly zero-knowledge. If the unit vector argument is knowledge sound and $\boldsymbol{q}^{+}$-MP commitment is binding, then the permutation argument is knowledge sound.*

```
Cgen^+(par):
  1. $(\mathsf{crs}_{pm}, \mathsf{td} = (x, \rho, \theta, \mathbf{K})) \leftarrow \mathsf{Cgen}^+_{pm}(\mathsf{par})$;
  2. $\mathsf{crs}_{\mathsf{CV}} \leftarrow ([x, \theta, \mathbf{K}]_1, [\{\theta^{2i-1}\}_{i=1}^n, \mathbf{K}]_2)$;
  3. $\mathsf{crs}^* \leftarrow [\{x^i\}_{i=1}^n]_1, [\{x^i\}_{i=1}^{2n}, \{K_1 x^i, K_2 \theta^{2i}\}_{i=1}^n, K_1 \rho]_2$; //only used in the
     security proof
  4. Return $(\mathsf{crs} = (\mathsf{crs}_{pm}, \mathsf{crs}_{\mathsf{CV}}, \mathsf{crs}^*), \mathsf{td})$;
P^+(par, crs, x = [C, C']_2, w = (A, t)):
  1. $\hat{\mathbf{r}} \leftarrow_\$ \mathbb{Z}_p^n$ such that $\sum_{i=1}^n \hat{r}_i = 0$; $[\hat{\mathbf{a}}]_2 \leftarrow \mathbf{A}^\top \cdot [\mathbf{q}^+(x)]_2 + \hat{\mathbf{r}}[1]_2$;
  2. $\pi_{pm} \leftarrow \mathsf{P}^+_{pm}(\mathsf{par}, \mathsf{crs}_{uv}, [\hat{\mathbf{a}}]_2, (\mathbf{A}, \hat{\mathbf{r}}))$;
  3. $r_t \leftarrow_\$ \mathbb{Z}_p$; $[s]_2 \leftarrow \mathbf{t}^\top [\mathbf{q}^+(x)]_2 + r_t [\hat{\rho}]_2$; $[\hat{\mathbf{c}}]_1 \leftarrow \hat{\mathbf{r}}^\top [\mathbf{C}]_1 + r_t \cdot \mathsf{pk}$;
  4. Return $\pi \leftarrow (([\hat{\mathbf{c}}]_1, [\hat{\mathbf{a}}, s]_2), \pi_{pm})$;
V^+(par, crs, x = [C, C']_2, π = (([ĉ]_1, [â, s]_2), π_pm)):
  1. $\mathsf{V}^+_{pm}(\mathsf{par}, \mathsf{crs}_{pm}, [\hat{\mathbf{a}}]_2, \pi_{pm}) \overset{?}{=} 1$;
  2. $[\mathbf{C}']_1^\top [\mathbf{q}^+(x)]_2 - [\mathbf{C}]_1^\top [\hat{\mathbf{a}}]_2 \overset{?}{=} \mathsf{pk} \cdot [s]_2 - [\hat{\mathbf{c}}]_1 [1]_2$;
```

Figure 11. Transparent shuffle argument $\mathcal{S}^+$

5.5. Shuffle Argument

Construction of the shuffle argument $\mathcal{S}^+$ in Fig. 11 is on the high level similar to $\mathcal{S}$. There are three big differences, however. Firstly, there is no same message argument as we merged it with the unit vector argument. Secondly, we inlined the consistency argument and do not treat it as a separate argument anymore. Thirdly, CRS will contain additional elements $\mathsf{crs}_{\mathsf{CV}}$ and crs^*. These are elements revealed by the CRS generation protocol that we mentioned at the beginning of this chapter. Elements in crs^* are a by-product that can be completely ignored by honest parties. For this reason, we did not consider them when counting CRS size in Table 1. The only reason why we mention crs^* is that it might be available to the adversary and this should be considered in the security proof. Elements in $\mathsf{crs}_{\mathsf{CV}}$ are going to be useful for the CRS verification algorithm CV which we will explain in the next section.

The soundness of $\mathcal{S}^+$ is based on a novel variation of KerMDH assumption that we call GapKerMDH.

Definition 19 (GapKerMDH Assumption). *n-GapKerMDH assumption holds relative to* BPgen*, if for any PPT adversary* A,

$$\Pr\left[\begin{array}{l} \mathsf{bp} \leftarrow \mathsf{BPgen}(1^\lambda), \theta \leftarrow_\$ \mathbb{Z}_p, [\boldsymbol{x}]_1 \leftarrow \mathsf{A}(\mathsf{bp}, [\theta]_1, \{[\theta^i]_2\}_{i=1}^{2n}) : \\ (1, \theta^2, \ldots, \theta^{2i}, \ldots, \theta^{2n}) \cdot \boldsymbol{x} = 0 \wedge \boldsymbol{x} \neq \boldsymbol{0} \end{array}\right] \approx_\lambda 0.$$

Notice that $(1, \theta^2, \ldots, \theta^{2i}, \ldots, \theta^{2n})$ has only even powers of θ. These gaps

```
$\mathsf{CV}(\mathsf{par}, \mathsf{crs})$:
  1. Assert that crs can be parsed according to Fig. 11 and elements belong to correct groups.
  2. $[\rho]_2 \overset{?}{\in} \mathbb{G}_2 \setminus [0]_2$; $[1]_T \overset{?}{=} [1]_1[1]_2$;
  3. For $z \in \{x, \theta, \rho, K_1, K_2\}$: $[z]_1[1]_2 \overset{?}{=} [1]_1[z]_2$;
  4. For $i \in [2..2n]$: $[1]_1[\theta^i]_2 \overset{?}{=} [\theta]_1[\theta^{i-1}]_2$;
  5. $[K_1^2]_1[1]_2 \overset{?}{=} [K_1]_1[K_2]_2$; $[1]_1[K_1^2\rho]_2 \overset{?}{=} [K_1^2]_1[\rho]_2$;
  6. $[1]_1[K_1K_2]_2 \overset{?}{=} [K_2]_1[K_1]_2$; $[K_1K_2]_1[1]_2 \overset{?}{=} [1]_1[K_1K_2]_2$;
  7. $[p_0(x)]_1[1]_2 \overset{?}{=} [1]_1[p_0(x)]_2$;
  8. For $i \in [1..n]$:
    (a) $[p_i(x)]_1[1]_2 \overset{?}{=} [1]_1[p_i(x)]_2$;
    (b) $[1]_1[K_1^2 p_i(x)]_2 \overset{?}{=} [K_1^2]_1[p_i(x)]_2 + [K_1K_2]_1[q_i^+(x)]_2$;
    (c) $[\rho]_1[((p_i(x) + p_0(x))^2 - 1)/\rho]_2 \overset{?}{=} [p_i(x) + p_0(x)]_1[p_i(x) + p_0(x)]_2 - [1]_T$;
  9. Return 0 if any of the equations is not satisfied and 1 otherwise.
```

Figure 12. CRS verification algorithm for $\mathcal{S}^+$

between powers are significant. If the vector would contain consecutive powers $(\dots, \theta^{2i}, \theta^{2i+1}, \dots)$ at any point, then A can output $[\mathbf{x}]_1 = [0, \dots, 0, \theta, -1, 0, \dots, 0]_1$ and break the assumption. We prove the following result in [AFK$^+$20].

Theorem 11. *If $(2n)$-SPDL assumption holds, then the n-GapKerMDH assumption holds in AGM.*

Now we can also state the security claim for the shuffle argument.

Theorem 12. *$\mathcal{S}^+$ is perfectly complete and perfectly zero-knowledge. If the permutation argument is knowledge sound and n-GapKerMDH assumption holds, then $\mathcal{S}^+$ has soundness.*

5.6. Subversion Zero-Knowledge

We modified CRS such that we can use the CRS generation protocol from Chapter 4. This guarantees soundness and zero-knowledge as long as at least 1 party in the protocol is honest. We now show how to achieve zero-knowledge even if all parties are malicious.

We construct a CRS verification algorithm CV that, roughly speaking, checks that all CRS elements used by the prover are well-formed. We do this with pairing-based verification equations described in Fig. 12. For convenience, we write elements such that they are already well-formed. In reality, we, of course, do not know it before executing the CV algorithm.

Let us consider line 4 in Fig. 12 as an example and we are going to denote the element that we verifying by X. Thus we have an equation $[1]_1[X]_2 \overset{?}{=} [\theta]_1[\theta^{i-1}]_2$ which is supposed to establish that $X = \theta^i$. We assume here that $[\theta]_1$ and $[\theta^{i-1}]_1$ are already known to be well-formed from previous equations. In such a case, the left-hand side of the equation is $[1]_1[X]_2 = [X]_T$ and the right-hand side of the equation is $[\theta]_1[\theta^{i-1}]_2 = [\theta^i]_T$. Thus clearly $X = \theta^i$. All the other elements can be shown to be well-formed using similar reasoning.

Finally, in [AFK$^+$20] we prove that such a CV algorithm guarantees subversion zero-knowledge under BDH-KE assumption. Knowledge assumption is needed since we need to extract the trapdoor from the subverter and provide it to the simulator.

Theorem 13. *If BDH-KE assumption holds, then $\mathcal{S}^+$ is Sub-ZK.*

6. CONCLUSIONS AND FUTURE WORK

In this thesis, we first proposed a NIZK shuffle argument in the CRS model which is secure without the RO model but is close in efficiency even to the best RO-based shuffle arguments. Then we constructed an efficient distributed CRS generation protocol for a special class of pairing-based NIZKs which also does not require the RO model. Finally, we proposed a slightly modified shuffle argument that has much lower trust requirements. In particular, we can apply the CRS generation protocol and this will guarantee security if at least one party in the protocol was honest. Zero-knowledge holds even without any trust assumptions.

We believe that the results of this thesis make RO-free shuffle arguments a viable option in real-life systems. The first argument is already part of the Zeus i-voting system which has been used for elections in many organizations [BCD$^+$19]. Our second argument, together with the CRS generation protocol and CRS verification algorithm, is planned to be integrated with Zeus in the future.

We mention one interesting direction for future work. Chase et al. [CKLM13b] proposed a controllably malleable shuffle argument that uses Groth and Lu's shuffle argument [GL07] as a building block. Essentially the idea is to let each mixer mall the proof of the previous mixer. Verifier only needs to check the final proof and proof size is almost independent of the number mixers. Since our shuffle argument is partly based on ideas of Groth and Lu, it seems plausible that similar malleability techniques apply here as well. However, our argument is several times more efficient in almost every parameter and could make such mix-nets much more efficient. Interestingly, malleability seems much more challenging in RO-based shuffles due to the non-malleability of hashing. Controlled malleability could be another big advantage for pairing-based shuffles, besides avoiding the RO model.

Still many open problems remain in this area. As a short term goal, it would be interesting to obtain as efficient non-interactive shuffle arguments under more standard assumptions. Faonio et al. [FFHR19] made recently good progress in this direction by achieving linear complexity (in all parameters) under SXDH assumption. However, there is still a noticeable gap in concrete efficiency between their work and the fastest shuffle arguments. In the long term, it is also important to obtain efficient post-quantum secure non-interactive shuffle arguments. Only very recently have we seen the first attempts in this area, e.g. [CMM17,Str19,CMM19].

BIBLIOGRAPHY

[Abe99] Masayuki Abe. Mix-networks on permutation networks. In Kwok-Yan Lam, Eiji Okamoto, and Chaoping Xing, editors, *ASIACRYPT'99*, volume 1716 of *LNCS*, pages 258–273. Springer, Heidelberg, November 1999.

[ABL+19a] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, Janno Siim, and Michal Zajac. DL-extractable UC-commitment schemes. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 385–405. Springer, Heidelberg, June 2019.

[ABL+19b] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, Janno Siim, and Michal Zajac. UC-secure CRS generation for SNARKs. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje eddine Rachidi, editors, *AFRICACRYPT 19*, volume 11627 of *LNCS*, pages 99–117. Springer, Heidelberg, July 2019.

[ABLZ17] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2017.

[AFK+20] Antonis Aggelakis, Prastudy Fauzi, Georgios Korfiatis, Panos Louridas, Foteinos Mergoupis-Anagnou, Janno Siim, and Michal Zajac. A non-interactive shuffle argument with low trust assumptions. In Stanislaw Jarecki, editor, *CT-RSA 2020*, volume 12006 of *LNCS*, pages 667–692. Springer, Heidelberg, February 2020.

[AHK20] Thomas Agrikola, Dennis Hofheinz, and Julia Kastner. On instantiating the algebraic group model from falsifiable assumptions. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 96–126. Springer, Heidelberg, May 2020.

[ALSZ18] Behzad Abdolmaleki, Helger Lipmaa, Janno Siim, and Michał Zając. On qa-nizk in the bpk model. Cryptology ePrint Archive, Report 2018/877, 2018. `https://eprint.iacr.org/2018/877`.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd FOCS*, pages 106–115. IEEE Computer Society Press, October 2001.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005.

[BBH+19] James Bartusek, Liron Bronfman, Justin Holmgren, Fermi Ma, and Ron D. Rothblum. On the (in)security of kilian-based SNARGs.

In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 522–551. Springer, Heidelberg, December 2019.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004.

[BCD$^+$19] Moritz Bartl, Pyrros Chaidos, George Danezis, Claudia Diaz, Harry Halpin, Helger Lipmaa, Panos Louridas, Benjamin Weggenmann, and Thomas Zacharias. Report on exploitation activities and updated plan for further exploitation, 2019. `https://panoramix-project.eu/wp-content/uploads/2019/03/D2.7.pdf` accessed in 10.03.2020.

[BCG$^+$14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *45th FOCS*, pages 186–195. IEEE Computer Society Press, October 2004.

[BD19] Razvan Barbulescu and Sylvain Duquesne. Updating key size estimations for pairings. *Journal of Cryptology*, 32(4):1298–1336, Oct 2019.

[BDG$^+$13] Nir Bitansky, Dana Dachman-Soled, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, Adriana López-Alt, and Daniel Wichs. Why "Fiat-Shamir for proofs" lacks a proof. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 182–201. Springer, Heidelberg, March 2013.

[BFF$^+$15] Gilles Barthe, Edvard Fagerholm, Dario Fiore, Andre Scedrov, Benedikt Schmidt, and Mehdi Tibouchi. Strongly-optimal structure preserving signatures from type II pairings: Synthesis and lower bounds. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 355–376. Springer, Heidelberg, March / April 2015.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988.

[BFS16] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, Heidelberg, December 2016.

[BG12] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280. Springer, Heidelberg, April 2012.

[BG18] Sean Bowe and Ariel Gabizon. Making groth's zk-SNARK simulation extractable in the random oracle model. Cryptology ePrint Archive, Report 2018/187, 2018. `https://eprint.iacr.org/2018/187`.

[BGM17] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050, 2017. `http://eprint.iacr.org/2017/1050`.

[BGR98] Mihir Bellare, Juan A. Garay, and Tal Rabin. Batch verification with applications to cryptography and checking. In Claudio L. Lucchesi and Arnaldo V. Moura, editors, *LATIN 1998*, volume 1380 of *LNCS*, pages 170–191. Springer, Heidelberg, April 1998.

[BNM$^+$14] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. Mixcoin: Anonymity for Bitcoin with accountable mixes. Cryptology ePrint Archive, Report 2014/077, 2014. `http://eprint.iacr.org/2014/077`.

[BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003.

[BP15] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 236–261. Springer, Heidelberg, November / December 2015.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[Cha81] David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[CKLM12] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 281–300. Springer, Heidelberg, April 2012.

[CKLM13a] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Succinct malleable NIZKs and an application to compact shuffles. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 100–119. Springer, Heidelberg, March 2013.

[CKLM13b] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Verifiable elections that scale for free. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 479–496. Springer, Heidelberg, February / March 2013.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 565–582. Springer, Heidelberg, August 2003.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

[CMM17] Nuria Costa, Ramiro Martínez, and Paz Morillo. Proof of a shuffle for lattice-based cryptography. In Helger Lipmaa, Aikaterini Mitrokotsa, and Raimundas Matulevičius, editors, *Secure IT Systems*, pages 280–296, Cham, 2017. Springer International Publishing.

[CMM19] Núria Costa, Ramiro Martínez, and Paz Morillo. Lattice-based proof of a shuffle. Cryptology ePrint Archive, Report 2019/357, 2019. `https://eprint.iacr.org/2019/357`.

[Dam92] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, Heidelberg, August 1992.

[Den02] Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 100–109. Springer, Heidelberg, December 2002.

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Heidelberg, December 2014.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Heidelberg, August 2012.

[EHK$^+$13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In

Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.

[FF18] Antonio Faonio and Dario Fiore. Optimistic mixing, revisited. Cryptology ePrint Archive, Report 2018/864, 2018. `https://eprint.iacr.org/2018/864`.

[FFHR19] Antonio Faonio, Dario Fiore, Javier Herranz, and Carla Ràfols. Structure-preserving and re-randomizable RCCA-secure public key encryption and its applications. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 159–190. Springer, Heidelberg, December 2019.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

[FL16] Prastudy Fauzi and Helger Lipmaa. Efficient culpably sound NIZK shuffle argument without random oracles. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 200–216. Springer, Heidelberg, February / March 2016.

[FLSZ17a] Prastudy Fauzi, Helger Lipmaa, Janno Siim, and Michal Zajac. An efficient pairing-based shuffle argument. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 97–127. Springer, Heidelberg, December 2017.

[FLSZ17b] Prastudy Fauzi, Helger Lipmaa, Janno Siim, and Michal Zajac. An efficient pairing-based shuffle argument. Cryptology ePrint Archive, Report 2017/894, 2017. `http://eprint.iacr.org/2017/894`.

[FLZ16] Prastudy Fauzi, Helger Lipmaa, and Michal Zajac. A shuffle argument secure in the generic model. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 841–872. Springer, Heidelberg, December 2016.

[FMM$^+$03] Jun Furukawa, Hiroshi Miyauchi, Kengo Mori, Satoshi Obana, and Kazue Sako. An implementation of a universally verifiable electronic voting scheme based on shuffling. In Matt Blaze, editor, *FC 2002*, volume 2357 of *LNCS*, pages 16–30. Springer, Heidelberg, March 2003.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M.

Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[FS01] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 368–387. Springer, Heidelberg, August 2001.

[Fuc18] Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347. Springer, Heidelberg, March 2018.

[GI08] Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 379–396. Springer, Heidelberg, April 2008.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pages 102–115. IEEE Computer Society Press, October 2003.

[GK16] Shafi Goldwasser and Yael Tauman Kalai. Cryptographic assumptions: A position paper. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 505–522. Springer, Heidelberg, January 2016.

[GL07] Jens Groth and Steve Lu. A non-interactive shuffle with pairing based verifiability. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 51–67. Springer, Heidelberg, December 2007.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, Heidelberg, August 2006.

[GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006.

[GR16] Alonso González and Carla Ràfols. New techniques for non-interactive shuffle and range arguments. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 16*, volume 9696 of *LNCS*, pages 427–444. Springer, Heidelberg, June 2016.

[Gro10] Jens Groth. A verifiable secret shuffle of homomorphic encryptions. *Journal of Cryptology*, 23(4):546–579, October 2010.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

[IKOS06] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *47th FOCS*, pages 239–248. IEEE Computer Society Press, October 2006.

[Kat] Katzenpost. `https://katzenpost.mixnetworks.org` accessed in 11.03.2020.

[KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 357–388. Springer, Heidelberg, August 2017.

[KW15] Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 101–128. Springer, Heidelberg, April 2015.

[LZ13] Helger Lipmaa and Bingsheng Zhang. A More Efficient Computationally Sound Non-Interactive Zero-Knowledge Shuffle Argument. *Journal of Computer Security*, 21(5):685–719, 2013.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, December 2005.

[MRV16] Paz Morillo, Carla Ràfols, and Jorge Luis Villar. The kernel matrix Diffie-Hellman assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 729–758. Springer, Heidelberg, December 2016.

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Heidelberg, August 2003.

[Nef01] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 116–125. ACM Press, November 2001.

[Pas13] Rafael Pass. Unprovable security of perfect NIZK and non-interactive non-malleable commitments. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 334–354. Springer, Heidelberg, March 2013.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *EUROCRYPT'95*, volume 921 of *LNCS*, pages 393–403. Springer, Heidelberg, May 1995.

[SP06] Krishna Sampigethaya and Radha Poovendran. A survey on mix networks and their secure applications. *Proceedings of the IEEE*, 94(12):2142–2181, Dec 2006.

[Str19] Martin Strand. A verifiable shuffle for the gsw cryptosystem. In Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala, editors, *Financial Cryptography and Data Security*, pages 165–180, Berlin, Heidelberg, 2019. Springer Berlin Heidelberg.

[TOR18] The tor project, 2018. `https://www.torproject.org` accessed in 27.11.2019.

[TPLT13] Georgios Tsoukalas, Kostas Papadimitriou, Panos Louridas, and Panayiotis Tsanakas. From helios to zeus. In *2013 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '13, Washington, D.C., USA, August 12-13, 2013*, 2013.

[TW10] Björn Terelius and Douglas Wikström. Proofs of restricted shuffles. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 10*, volume 6055 of *LNCS*, pages 100–113. Springer, Heidelberg, May 2010.

ACKNOWLEDGEMENT

Most of all, I would like to thank my supervisor, Helger Lipmaa, for giving me an opportunity to join his group and for showing me how to be a researcher. I also thank everyone else that was part of the cryptographic protocols research group in Tartu. Especially, Behzad Abdolmaleki, Karim Baghery, Michał Zając, and Prastudy Fauzi. I thank all the other numerous co-authors that I had over the years. In particular Thomas Zacharias, whom I learned a lot. I thank Sven Heiberg and Jan Willemson for giving me an opportunity to work on the more applied side of cryptography during my time in STACC. I thank the GRNET team (lead by Panos Louridas) for implementing several of our protocols. I also thank the internal reviewer Dominique Unruh and the external pre-reviewers Eran Tromer and Dario Fiore for giving useful feedback that helped to improve the final version of this work.

Throughout my studies, I have been funded by European Union's Horizon 2020 research and innovation program under grant agreement No 653497 (project PANORAMIX) and No. 780477 (project PRIViLEDGE), by the Estonian Research Council grant (PRG49), and by European Regional Development Fund under the grant no EU48684.

SUMMARY IN ESTONIAN

Mitte-interaktiivsed segamise nullteadmustõestused

Demokraatliku riigi toimimiseks on turvaline hääletussüsteem esmatähtis. Viimastel aastatel on Eesti ja Šveits kasutusele võtnud e-hääletussüsteemid ja mitmed teised riigid on sarnaseid süsteeme kas proovinud kasutada (nt Norra ja Austraalia) või plaanivad neid kasutada tulevikus (nt Leedu ja Venemaa). E-hääletus on korraldajale odavam ning valijale mugavam, mis võib tõsta valimisaktiivsust. Samas kaasnevad sellega ka paljud uued turvariskid ja tehnilised väljakutsed.

Võtame näiteks häälte segamise protseduuri läbi hääletuskasti raputamise, mis peaks kaotama seose hääletaja ja tema hääle vahel. Seda on väga lihtne teostada paberhäältega, kuid pole kaugeltki ilmne, kuidas saavutada sama tulemus digitaalsete krüpteeritud häältega. Kes peaks teostama segamise? Kuidas me saame kindlad olla, et segamine tehti korrektselt? Kas hääle omanikku on tõesti võimatu tuvastada?

Selle ülesande lahendamiseks saab kasutada hajussüsteemi, mida kutsutakse mix-netiks. Iga server mix-netis segab häälte krüptogrammid ehk paigutab need juhuslikku järjekorda ja muudab nende välimust. Kui vähemalt üks serveritest on aus, siis on arvutuslikult raske seostada krüptogrammi ja tema omanikku. Seda juhul, kui iga server lisaks ka tõestab, et ta segas hääled korrektselt. Tõestus peaks olema raskesti võltsitav ja ei tohi avalikustada rohkem informatsiooni kui see, et segamine toimus korrektselt. Vastavate omadustega tõestusi nimetatakse segamise nullteadmustõestusteks ja selles töös uurime, kuidas neid konstrueerida. Erinevalt paljudest eelnevatest töödest, ei vaja meie konstruktsioon idealiseeritud juhuslikku mudelit, kus räsifunktsiooni käsitletakse juhusliku funktsioonina. Me näitame, et ka ilma juhusliku oraakli mudelita on võimalik konstrueerida segamise nullteadmustõestusi, mis on piisavalt tõhusad suuremahuliste valimiste jaoks.

Antud töö koosneb kolmest eraldiseisvast osast. Esiteks konstrueerime seni kõige tõhusama segamise nullteadmustõestuse, mis ei vaja juhusliku oraakli mudelit. Selleks kombineerime mitmeid hiljutisi saavutusi nullteadmustõestuste valdkonnast. Lisaks implementeerime tõestuse ja näeme, et see on piisavalt kiire praktikas kasutamiseks: väga tagasihoidlikul tarkvaral võtab 100 000 krüptogrammi segamine ja tõestamine aega alla 1,5 minuti ning ka tõestuse verifitseerimiseks kulub umbes sama palju aega. Kahjuks kehtivad turvagarantiid vaid juhul, kui tõestajal ja verifitseerijal on ligipääs usaldatud viitesõnele.

Teiseks uurime, kuidas sarnastes nullteadmustõestustes usaldust vähendada. On olemas turvalisi ühisarvutussüsteeme, mis suudavad arvutada viitesõnesi paljudele erinevatele nullteadmustõestustele. Kahjuks kasutavad ka need süsteemid juhusliku oraakli mudelit. Me täiendame ühte sellist ühisarvutusprotokolli ning muuhulgas kaotame juhuslikkuse oraakli nõude ja tõestame protokolli turvalisust väga tugevas täieliku koosluskindluse mudelis.

Kolmandaks modifitseerime eelnevat segamise nullteadmustõestust viisil, et

see toimiks koos ühisarvutuse protokolliga. See kindlustab, et nullteadmustõestus on turvaline seni, kuni vähemasti üks osapool ühisarvutuses oli aus. Lisaks astume sammu võrra kaugemale ja näitame, kuidas säilitada privaatsus ka siis, kui kõik osapooled ühisarvutuses olid pahatahlikud. Samuti lihtsustame esialgset konstruktsiooni ja tõestame protokolli turvalisust nõrgematel krüptograafilistel eeldustel.

PUBLICATIONS

CURRICULUM VITAE

Personal data

Name:	Janno Siim
Birth:	March 18th, 1992
Citizenship:	Estonian
Languages:	Estonian, English
E-mail:	janno.siim@ut.ee

Education

2016–...	University of Tartu, PhD candidate in Computer Science
2014–2016	University of Tartu, MSc in Computer Science
2011–2014	University of Tartu, BSc in Computer Science
2008–2011	Kuressaare Secondary School
1999–2008	Kaarma Primary School

Employment

01.09.2017–31.08.2018	STACC, Scientist
07.09.2016–31.08.2017	STACC, Junior Researcher
15.09.2016–31.12.2020	University of Tartu, Junior Research Fellow in Cryptography
01.05.2016–31.08.2016	University of Tartu, Research Project Specialist

Scientific work

Main fields of interest:

- Cryptography
- Zero-knowledge proofs
- Electronic voting

ELULOOKIRJELDUS

Isikuandmed

Nimi:	Janno Siim
Sünniaeg:	18. märts 1992
Kodakondsus:	eestlane
Keelteoskus:	eesti, inglise
E-post:	janno.siim@ut.ee

Haridus

2016–...	Tartu Ülikool, doktoriõpe informaatikas
2014–2016	Tartu Ülikool, magistrantuuriõpe informaatikas
2011–2014	Tartu Ülikool, bakalaureuseõpe informaatikas
2008–2011	Kuressaare Gümnaasium
1999–2008	Kaarma Põhikool

Teenistuskäik

01.09.2017–31.08.2018	STACC, teadur
07.09.2016–31.08.2017	STACC, nooremteadur
15.09.2016–31.12.2020	Tartu Ülikool, krüptograafia nooremteadur
01.05.2016–31.08.2016	Tartu Ülikool, teadusprojekti spetsialist

Teadustegevus

Peamised uurimisvaldkonnad:

- Krüptograafia
- Nullteadmustõestused
- Elektrooniline hääletamine

DISSERTATIONES INFORMATICAE PREVIOUSLY PUBLISHED IN DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS