Poster: Efficient Three-Party Shuffling Using Precomputation

Andreas Brüggemann
brueggemann@encrypto.cs.tu-darmstadt.de
Technical University of Darmstadt

Thomas Schneider
schneider@encrypto.cs.tu-darmstadt.de
Technical University of Darmstadt

Ajith Suresh
suresh@encrypto.cs.tu-darmstadt.de
Technical University of Darmstadt

Hossein Yalame
yalame@encrypto.cs.tu-darmstadt.de
Technical University of Darmstadt

ABSTRACT

In this paper, we revisit the problem of secure shuffling in a three-server setting with an honest majority. We begin with the recent work of Araki. et al. (CCS'21) and use precomputation to improve the communication and round complexity of the online phase of their shuffle protocol. Our simple yet effective shuffling method is not limited to three parties and can be used in a variety of situations. Furthermore, the design of our solution allows for fine tuning to achieve improved efficiency based on the underlying application's parameters. Our protocols are initially presented with semi-honest security and then extended to support malicious corruption.

CCS CONCEPTS

• **Security and privacy → Cryptography**; **Privacy-preserving protocols**.

KEYWORDS

Multi-party Computation; Secure Shuffle; 3PC; Preprocessing

ACM Reference Format:
Andreas Brüggemann, Thomas Schneider, Ajith Suresh, and Hossein Yalame. 2022. Poster: Efficient Three-Party Shuffling Using Precomputation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), November 7–11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3548606.3563511

1 INTRODUCTION

Shuffling a data set while keeping the contained data private is of particular interest for a wide range of applications, including highly scalable message passing graph algorithms (for example, contact tracing) [2], metadata-hiding communication [6], and computing database joins [8]. While mixnet-based approaches were used earlier to shuffle private data, there has been a shift in recent years toward using secure multi-party computation (MPC) techniques due to practical efficiency. MPC allows multiple parties to compute a function on their private inputs so that no party learns anything other than its input and output. Several recent works used MPC with a small number of parties to achieve practically efficient and scalable solutions [1, 4, 10]. Furthermore, most of them relied on the preprocessing model to offload the expensive tasks to an input-independent preprocessing phase [4, 7, 9]. This will aid in obtaining a fast online time and better throughput, both of which are critical when considering real-world applications. Our work aims to reduce the online costs of existing shuffling protocols in the three-server honest majority setting (3PC), where only one server can be corrupt.

The shuffling protocol by Araki et al. [2] forms the state-of-the-art in the 3PC setting. Their approach makes use of secret share replication and employs a pair-wise shuffle method in which each pair of servers performs a joint shuffle of the secret using a permutation unknown to the third server. While this approach results in the communication of six messages over three online rounds, they proposed an optimization for the semi-honest case in which the same could be accomplished in two rounds with four messages. An easy adaptation of their approach to the preprocessing model is to replace the underlying 3PC scheme of [1] with ASTRA [4], which will aid in moving communication of two of the four messages to the preprocessing. However, the modified protocol still necessitates two rounds of online interaction.

Clarion [6] recently proposed a shuffle protocol for anonymous communication in the 3PC setting. However, their configuration is slightly different, and they use the third server as a helper in the preprocessing to aid in secure server shuffling between two servers. In the semi-honest case, we observe that it is easy to modify the Clarion protocol to obtain a 3PC shuffling scheme. However, the modification will necessitate an additional two-message communication, worse than [2] in the preprocessing model. Furthermore, the solution necessitates two rounds in the online phase.

Our contributions. We propose protocols for shuffling in the 3PC setting with an improved online phase leveraging precomputation. Our contributions are as follows:

– In the semi-honest setting, we propose a 3PC shuffle protocol that reduces the online costs of [2] by 2×. We present two variants with varying trade-offs in computation and communication. Our method is of independent interest and can be used to obtain better constructions in other settings (§2).

– We extend our protocols to handle malicious corruption while maintaining the goal of a better online phase. Our solution significantly improves the online round complexity of the malicious shuffle in [2] (§2).

– We provide insights on how to apply our methods in the context of secure graph analysis, as proposed in [2]. We observe that our solutions enjoy further improvements over [2] when used in scenarios where the same shuffle is used multiple times (§2).

CCS '22, November 7–11, 2022, Los Angeles, CA, USA

ACM ISBN 978-1-4503-9450-5/22/11.
https://doi.org/10.1145/3548606.3563511

2 OUR APPROACH TOWARDS SHUFFLING

Consider a table $\mathsf{T} \in \{0,1\}^{n \times m}$ with n rows, each containing m-bit entries, that is secret-shared among three servers S_0, S_1 and S_2. The goal is to permute the rows of T by a random permutation π that no single server knows, to obtain shuffled table $\pi(\mathsf{T})$. First, observe that for every permutation π, there exists a row permutation matrix $M_\pi \in \{0,1\}^{n \times n}$, so that shuffling T can be expressed as matrix multiplication of the form $M_\pi \odot \mathsf{T} = \pi(\mathsf{T})$. Moreover, the matrix multiplication can be performed by evaluating nm dot products with vectors of length n, one for each result entry. We now explain how to obtain the permutation matrix and securely evaluate the dot products using the three-party protocol of ASTRA [4]. It is worth noting that the approach of multiplying with a permutation matrix is not limited to our three-server scenario. Furthermore, the approach enables different levels of security to be achieved by simply employing appropriate MPC schemes to generate the permutation matrix and perform the dot product protocol.

Semi-Honest Shuffling. Consider the following (insecure) approach: Server S_0 chooses a random permutation π_a, generates the corresponding row permutation matrix M_{π_a}, and secret-shares it among all servers. The servers can then simply compute the secret shares of $M_{\pi_a} \odot \mathsf{T}$ in ASTRA using the dot product protocol. Unfortunately, this technique is insufficient for shuffling because it simply uses a permutation known to S_0. To overcome this problem, we can let S_1 and S_2 apply a second permutation π_b to $M_{\pi_a} \odot \mathsf{T}$ that is not known to S_0. Thus, the result is a shuffled table $\pi_b(M_{\pi_a} \odot \mathsf{T})$ which exactly is $\pi(\mathsf{T})$ for $\pi := \pi_b \circ \pi_a$. Note that π is even random if only one of the permutations π_a, π_b is, and no single server knows anything about π. The remaining challenge is to efficiently compute $M_{\pi_a} \odot \mathsf{T}$ so that S_1 and S_2 can still apply π_b without leaking it to S_0. We take advantage of the asymmetry of the server roles in ASTRA, where S_0 acts as a helper and S_1 and S_2 act as evaluators, and our solution incurs no additional communication over the dot product to incorporate π_b. The details of the modified dot product protocol, in the context of our shuffling approach, are presented next.

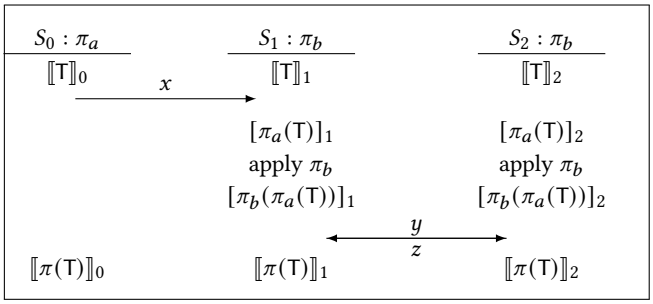

Figure 1: High-level overview of our 3-party shuffle. We denote XOR shares of x (among S_1 and S_2) by $[x]$, and ASTRA shares by $[\![x]\!]$.

Figure 1 provides a high-level overview of the resulting protocol. The protocol is modified to generate an XOR sharing of $M_{\pi_a} \odot \mathsf{T} = \pi_a(\mathsf{T})$ between S_1 and S_2, i.e., S_1 holds $[\pi_a(\mathsf{T})]_1$ and S_2 holds $[\pi_a(\mathsf{T})]_2$ s.t. $[\pi_a(\mathsf{T})]_1 \oplus [\pi_a(\mathsf{T})]_2 = \pi_a(\mathsf{T})$. Each of S_1 and S_2 can then generate the replicated sharing of their shares as per ASTRA's sharing semantics. The servers can locally XOR the resulting shares to obtain the final result. Note that this modification has the same communication cost as the dot product protocol in ASTRA. Most importantly, it enables S_1 and S_2 to permute their XOR sharings by π_b prior to generating the ASTRA sharing without involving S_0. As a result, they can convert their XOR share of $\pi_a(\mathsf{T})$ to that of $\pi_b(\pi_a(\mathsf{T}))$, resolving the challenge. It remains to show how the permutation matrix M_{π_a} of dimensions $n \times n$ can be efficiently shared by S_0. Sharing each entry requires quadratic communication in n and forms **Variant I** of our scheme. Since each row of M_{π_a} contains a single 1 entry and the rest are zeroes, we can use the sparsity of the matrix to reduce communication for secret-sharing.

In detail, we improve communication while increasing computation and use distributed point functions (DPFs) [3] to enable S_0 to generate secret shares of the matrix M_{π_a}. In concrete terms, S_0 generates DPF keys for each row of M_{π_a} based on π_a and sends these keys to S_1 and S_2, who can then use them to obtain an XOR sharing of the matrix M_{π_a} amongst themselves. We also take advantage of the fact that this XOR sharing between S_1 and S_2, combined with S_0's knowledge of it, enables the servers to obtain an ASTRA sharing of M_{π_a} without additional communication. Using the state-of-the-art DPF approach by Boyle et al. [3] results in keys $\mathsf{k}^n_{\mathsf{DPF}}$ of size $\approx \log_2(n/\kappa) \cdot \kappa$ for computational security parameter κ. As a result, we can improve the communication of sharing M_{π_a} from $O(n^2)$ to $O(n \log n)$, resulting in **Variant II** of our scheme. In terms of computation, this approach requires $O(n \log n)$ PRG operations (e.g., by utilizing AES) for each row of M_{π_a}. This could be reduced to $O(n)$ by increasing the key size by only $\approx \log \log n$ bits [3]. Since DPFs are treated as a black box, any future improvements to it could directly benefit our scheme. Furthermore, since these permutation matrices are input-independent, all of these operations can be performed beforehand during a precomputation phase, resulting in a fast online phase, which is the goal of our work.

Table 1: Comparison of semi-honest three-party shuffling schemes. The costs are reported for shuffling a table of n rows with m bits each. $|\mathsf{k}^n_{\mathsf{DPF}}|$ denotes the DPF key size.

Shuffle Protocol	Communication		Online Rounds
	Preprocessing	Online	
Araki et al. [2]	—	$4nm$	2
Araki et al. [2][a]	$2nm$	$2nm$	2
Clarion [6][b]	$2nm$	$3nm$	2
This-I	$n^2 + nm$	$2nm$	**1**
This-II	$2n \cdot \lvert\mathsf{k}^n_{\mathsf{DPF}}\rvert + nm$	$2nm$	**1**

[a] adapted to the precomputation model using [4]
[b] modified to the 3-party setting using [4]

Table 1 summarises our results. We improve upon the online round complexity of existing solutions. In cases where $m \geq n$, our solutions outperform the state-of-the-art in terms of overall communication as well. Furthermore, **Variant II** even outperforms the state-of-the-art's total communication for the more general case where $m \geq 2\kappa \cdot \log_2 n$. To summarize, we improved existing solutions' online costs (communication and rounds) while also providing superior total communication for sufficiently large values of m.

Extension to Malicious Security. We provide insights into improving the security of our semi-honest scheme to malicious security in this section. Due to space constraints, we avoid detailed technical discussions.

For the malicious case, we use the 3-party protocol of BLAZE [10, 11]. In contrast to ASTRA, the dot product protocol in BLAZE does not provide a way to include a second permutation (π_b) for free, as in §2. Instead, we let S_1 choose a random permutation π_b and share the corresponding row permutation matrix M_{π_b} as previously done for S_0 and M_{π_a}. Now, it holds that $\pi(\mathsf{T}) = \pi_b((\pi_a(\mathsf{T})) = M_{\pi_b} \odot M_{\pi_a} \odot \mathsf{T}$. Due to the associativity of matrix multiplication, $M_{\pi_b} \odot M_{\pi_a} \odot \mathsf{T}$ can be computed in two ways. In the first method (**Variant III**), $M_{\pi_b} \odot M_{\pi_a}$ is computed first, then multiplied by T. In the second method (**Variant IV**), we compute $M_{\pi_a} \odot \mathsf{T}$ and multiply it by M_{π_b} to get the result. Because of the difference in matrix dimensions, the communication costs of the two approaches differ. While **Variant III** can shift most of the computation to preprocessing, **Variant IV** requires two sequential matrix multiplications in the online phase, doubling the online rounds. On the other hand, **Variant IV** has better communication in the preprocessing than **Variant III** for $m \leq n$. Regarding the generation of permutation matrices, we use verifiable DPFs [5] to ensure that each row of the permutation matrix is correctly generated. However, this is insufficient for our case; we must also ensure that each column of the permutation matrix contains exactly one 1 and the rest are zeroes. For this, once the secret shares of the permutation matrix are generated, servers locally perform an XOR of all the rows of the matrix to obtain a single vector in secret shared form. This vector is publicly opened and servers abort if any of the entries is a zero.

Table 2: Costs for shuffling a table of n rows with m bits each in three-party setting with malicious security.

Shuffle Protocol	Communication		Online Rounds
	Preprocessing	Online	
[2][a]	—	$> 24\sigma n + 6nm$	$> 9 + 3\log\sigma$
This-III	$\approx 3n^2 + 3nm + 4n \cdot \lvert \mathsf{k}^n_{\mathsf{VDPF}} \rvert$[b]	$3nm$	1
This-IV	$\approx 6nm + 4n \cdot \lvert \mathsf{k}^n_{\mathsf{VDPF}} \rvert$	$6nm$	2

[a] for statistical security parameter σ [b] verifiable DPFs key

Table 2 summarises our results for the malicious setting. In this case, we significantly improve upon the online round complexity of the malicious shuffling protocol by Araki et al. [2]. This is primarily due to the requirement in [2] for a round expensive shuffle verification after each shuffle-pair operation. On the other hand, the underlying three-party protocol ensures the malicious security of our approach if indeed the permutation matrices are correctly generated. The communication analysis is similar to that of the semi-honest case. However, for large values of n, our approach has worse overall communication than [2]. We are currently investigating the details of DPF constructions to reduce the cost of generating the permutation matrix. Simultaneously, we are investigating the feasibility of using homomorphic encryption techniques to reduce the cost of permutation matrix generation.

Application to Graph Analysis. In this section, we look at how our method improves on the shuffle-based graph analysis approach by Araki et al. [2]. Secure shuffle is used as a building block in their work to perform the topology-hiding computation of message-passing graph algorithms[1]. At a high level, their algorithm runs for k iterations, with each iteration passing messages from nodes to neighbours. Each iteration uses two secure shuffles, one in the forward direction and one in the backward direction (cf. Fig. 2 of [2]). However, the same shuffles are used in all k iterations. While [2] requires communication corresponding to a shuffle in each iteration, our approach requires the corresponding permutation matrix to be generated only once and used in *all* the iterations. This will result in much greater improvements over [2] than we observed in the previous sections. The results are summarized in Table 3. This demonstrates that our approach is an excellent candidate for graph algorithms such as BFS traversal up to high or unlimited depth, and the Bellman-Ford algorithm with $k = n$ iterations.

[1] In [2], $m \geq \log_2 n$ for message passing graph algorithms.

Table 3: Costs for shuffling a table of n rows with m bits each over k iterations in the three-party semi-honest setting. The same shuffle is applied k times.

Shuffle Protocol	Communication		Online Rounds
	Preprocessing	Online	
[2]	—	$4knm$	$2k$
This-II	$2n \cdot \lvert \mathsf{k}^n_{\mathsf{DPF}} \rvert + knm$	$2knm$	k

Another thing to keep in mind is that the graph algorithm in [2] requires a reverse shuffle, i.e., permute according to π^{-1}. While this is straightforward for our malicious protocols, the semi-honest variants require special consideration. This results in an additional $2knm$ communication in preprocessing and thus has slightly worse overaall communication than [2]. We do, however, retain the advantage of improved communication and rounds during the online phase. We leave the problem of improving the overall communication to future work.

Acknowledgements. This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 850990 PSOTI). It was co-funded by the Deutsche Forschungsgemeinschaft (DFG) within SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/ 251805230, and by the BMBF and the HMWK within ATHENE.

REFERENCES

[1] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. 2016. High-Throughput Semi-Honest Secure Three-Party Computation with an Honest Majority. In *ACM CCS*.
[2] Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. 2021. Secure Graph Analysis at Scale. In *ACM CCS*.
[3] Elette Boyle, Niv Gilboa, and Yuval Ishai. 2016. Function Secret Sharing: Improvements and Extensions. In *ACM CCS*.
[4] Harsh Chaudhari, Ashish Choudhury, Arpita Patra, and Ajith Suresh. 2019. ASTRA: High Throughput 3PC over Rings with Application to Secure Prediction. In *CCSW@CCS*.
[5] Leo de Castro and Antigoni Polychroniadou. 2022. Lightweight, Maliciously Secure Verifiable Function Secret Sharing. In *EUROCRYPT*.
[6] Saba Eskandarian and Dan Boneh. 2022. Clarion: Anonymous Communication from Multiparty Shuffling Protocols. In *NDSS*.
[7] Nishat Koti, Mahak Pancholi, Arpita Patra, and Ajith Suresh. 2021. SWIFT: Superfast and Robust Privacy-Preserving Machine Learning. In *USENIX Security*.
[8] Payman Mohassel, Peter Rindal, and Mike Rosulek. 2020. Fast Database Joins and PSI for Secret Shared Data. In *ACM CCS*.
[9] Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. 2021. ABY2.0: Improved Mixed-Protocol Secure Two-Party Computation. In *USENIX Security*.
[10] Arpita Patra and Ajith Suresh. 2020. BLAZE: Blazing Fast Privacy-Preserving Machine Learning. In *NDSS*.
[11] Ajith Suresh. 2021. MPCLeague: Robust MPC Platform for Privacy-Preserving Machine Learning. *CoRR* abs/2112.13338 (2021).