A Fast, Practical and Simple Shortest Path Protocol for Multiparty Computation

Abdelrahaman Aly[1,3(✉)] and Sara Cleemput[2,3]

[1] Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi, UAE
[2] Emweb bv, Herent, Belgium
[3] imec-COSIC,KU Leuven, Leuven, Belgium
Abdelrahaman.aly@tii.ae

Abstract. We present a simple and fast protocol to securely solve the (single source) Shortest Path Problem, based on Dijkstra's algorithm over Secure Multiparty Computation. Our protocol improves the current state of the art by Aly et al. [FC 2013 & ICISC 2014] and can offer perfect security against both semi-honest and malicious adversaries. Furthermore, it is the first data oblivious protocol to achieve quadratic complexity in the number of communication rounds. Moreover, our protocol can be easily adapted as subroutine in other combinatorial mechanisms. Our focus is usability; hence, we provide an open source implementation and exhaustive benchmarking under different adversarial settings and players setups.

Keywords: Shortest path problem · Secure multi-party computation

1 Introduction

The (Single Source) Shortest Path problem (SPP), i.e. computing the shortest path between a source and all other vertices in a graph, is a commonly used subroutine in commercial applications. In many of these settings, data related to the computation of the problem such as elements of its configuration, graph topology or associated weights, can be considered private. Real life examples include telecommunication networks for banking or restricted topology combinatorial auctions, among others. In such environments, different parties could gain a competitive advantage by obtaining privately held information. Therefore, mechanisms to ensure secrecy, correctness and fairness are required.

In this work we introduce a Secure Multiparty Computation (MPC) data-oblivious protocol to securely solve the single source SPP. Just like in previous works, namely Aly et al. [2,4], we propose a data oblivious version of Dijkstra's algorithm, compatible with MPC. We consider all information related to the graph (aside from the number of vertices) to be privately held. The result of our computation is the length of the path and/or the path composition; the parties can then decide whether these are disclosed. Moreover, it can offer perfect

V. Atluri et al. (Eds.): ESORICS 2022, LNCS 13556, pp. 749–755, 2022.
https://doi.org/10.1007/978-3-031-17143-7_37

security[1] and its multiplicative complexity i.e. round complexity, is one order of magnitude lower than the current state of the art [2,4].

1.1 Related Work

Aly et al. [2,4] have introduced several data-oblivious protocols to solve the SPP, including adaptations of Dijkstra. However, their complexity bound on the number of sequential multiplications is cubic, whereas we only require a quadratic number of such multiplications. Brickell and Shmatikov [8] introduced a protocol for the SPP in a two-party setting against semi-honest adversaries. In contrast, our solution is not limited to the two-party case and also provides security against active adversaries. The Breadth-First-Search (BFS) proposed by Blanton et al. [7] provides complexity bounds for a special case of the SPP i.e. non-weighted graph. Conversely, we consider the general case where the graph is weighted. Furthermore, Keller and Scholl [17] implemented Dijkstra's algorithm using Oblivious RAM (ORAM) based data-structures matching the $\mathcal{O}(|V|^2)$ complexity of the original algorithm. However, their results show that for certain graph sizes, Aly et al. [2] can out-perform their ORAM-based implementation, as ORAM's intrinsic overhead exceeds any asymptotic advantage.

1.2 Notation and Security

We make use of the square brackets notation for secret shared values e.g. $[\![x]\!]$. Furthermore, we consider all inputs to be elements of $\mathbb{Z}_q$, where q is a sufficiently large[2] prime or RSA modulus. Complexity is measured in terms of round complexity (multiplicative depth or latency) of the whole protocol. Vectors and matrices are represented by capital letters e.g. E, where $|E|$ denotes its size. Finally, some common encapsulations used throughout our protocols are denoted as follows:

- $[\![z]\!] \leftarrow_{[\![c]\!]} [\![x]\!] : [\![y]\!]$ is the conditional operator. It can be seen as an arithmetic replacement for the `if` branching instruction. Here, $[\![c]\!]$ represents a selection bit and $[\![z]\!]$ takes the value of $[\![x]\!]$ if $[\![c]\!] \stackrel{?}{==} 1$ and $[\![y]\!]$ otherwise. This simple construction requires only one communication round i.e. $[\![c]\!] \cdot ([\![x]\!] - [\![y]\!]) + [\![y]\!]$.
- `exchange`$(i, j, [\![X]\!])$ swaps the elements in the i-th and j-th position of vector X. This operation is not cryptographic in nature.

Security of MPC protocols is typically defined in the context of simulation under the UC framework [9,10]. To simplify the analysis, we abstract the required MPC ideal functionality as an **arithmetic black box** or $\mathcal{F}_{ABB}$. Initially introduced by Damgård and Nielsen [13], it can be extended to support other ideally modeled functionality e.g. secure comparisons. We offer a revision of our $\mathcal{F}_{ABB}$, including corresponding UC secure realizations in Table 1. We proceed to define security as follows:

[1] From an Ideal perspective, and under the adequate setting i.e. honest majority. In practice, the protocol is as secure as the underlying MPC realization.

[2] It can instantiate the underlying MPC protocol.

Table 1. Secure Arithmetic operations provided by the $\mathcal{F}_{ABB}$.

Functionality	Description	Rounds	Prot.
$x \leftarrow [\![x]\!]$	Opening secret field element	1	e.g. [16,19]
$[\![x]\!] \leftarrow x$	Storing public input in a secret field element	1	e.g. [16,19]
$[\![z]\!] \leftarrow [\![x]\!] + [\![y]\!]$	Addition: of secret inputs	0	e.g. [16,19]
$[\![z]\!] \leftarrow [\![x]\!] + y$	Addition: (mixed) secret and public inputs	0	e.g. [16,19]
$[\![z]\!] \leftarrow [\![x]\!] \cdot [\![y]\!]$	Multiplication: of secret inputs	1	e.g. [16,19]
$[\![z]\!] \leftarrow [\![x]\!] \cdot y$	Multiplication: (mixed) secret and public inputs	0	e.g. [16,19]
—Complex Building Blocks—			
$[\![z]\!] \leftarrow [\![x]\!] \overset{?}{<} y[\![y]\!]$	Inequality Test: secret inputs	4–6	e.g. [3,11]
$[\![E]\!] \leftarrow \texttt{permute}([\![E]\!])$	secret random permutation of $[\![E]\!]$	approx $n \cdot \log(n)$	e.g. [12,15,18]

Definition 1. Let $\pi_{\texttt{SP}}$ be a real protocol implemented in a multiparty setting. We say $\pi_{\texttt{SP}}$ is UC-secure if, for any adversary $\mathscr{A}$, there exists a simulator $\mathscr{S}$ such that the $\texttt{VIEW}_{\pi}(P_i)$ of any party P_i interacting with the environment $\mathscr{Z}$, cannot be distinguished (with non-negligible probability) between the real protocol $\pi_{\texttt{SP}}$ and the ideal functionality $\mathcal{F}_{SP}$.

2 Privacy Preserving Single Source SPP

Let $G = (V, E)$ be a directed graph without negative cycles where V is the set of vertices and E is the set of edges. Furthermore, G is represented as a weighted adjacency matrix $[\![U]\!]$ where $[\![U]\!]_{ij}$ is the weight of edge (i, j) $\forall (i, j) \in E$. The intuition underlying our protocol is as follows: $[\![U]\!]$ is obliviously permuted before protocol execution. We then assign temporary labels to each vertex in G (i.e. each row in $[\![U]\!]$). Our protocol then proceeds to identify the most suitable vertex to explore. However, unlike other works in the field, given the permutation, we are able to open the next vertex temporary label and directly explore it. Note that the label itself does not convey any information other than the position of the row in the now permuted matrix $[\![U]\!]$.

Complexity: Our protocol requires $\mathcal{O}(|V|^2 \cdot log(|V|))$ secure multiplications (amount of work). Such multiplications can be parallelized achieving $\mathcal{O}(|V|^2)$ rounds of communication. Furthermore, Protocol 1 contains two additional multiplications in line 17 and 18, which can also be parallelized. The `exchange` operation does not influence the complexity of the protocol, as it is done over publicly available information.

Security Analysis: Our protocol does not disclose any private information during its execution. More precisely, the call to `open`($[v]$) (in line 12 of Protocol 1)

Protocol 1: Optimized Non-Disclosure Dijkstra Protocol ($\pi_{\mathtt{SP}}$)

Input: secret shared edge weights $[U]_{i,j}$ for $i, j \in \{1, ..., |V|\}$, encoding vector $[S]$ where $S_i = 0$ if $i \neq s$ (s being the source vertex) and 1 otherwise.
Output: The vector of predecessors α and the vector of distances $[D]$.

for $i \leftarrow 1$ **to** $|V|$ **do**
  $[\![\alpha]\!]_i \leftarrow i$; $[\![D]\!]_i \leftarrow_{[\![S_i]\!]} [\![0]\!] : [\![\top]\!]$; $[\![P]\!]_i \leftarrow [\![i]\!]$;
end
$([\![P]\!], [\![D]\!], [\![U]\!]) \leftarrow \mathtt{permute}([\![P]\!], [\![D]\!], [\![U]\!])$;
for $i \leftarrow 1$ **to** $|V|$ **do**
  $[\![d']\!] \leftarrow [\![\top]\!]$;
  for $j \leftarrow |V|$ **to** i **do**
    $[\![c]\!] \leftarrow [\![D]\!]_j \overset{?}{<} [\![d']\!]$;
    $[\![v]\!] \leftarrow_{[\![c]\!]} j : [\![v]\!]$;
    $[\![d']\!] \leftarrow_{[\![c]\!]} [\![D]\!]_j : [\![d']\!]$;
  end
  $v \leftarrow \mathtt{open}([\![v]\!])$;
  $\mathtt{exchange}(i, v, [\![P]\!], [\![D]\!], [\![U]\!])$;
  for $j \leftarrow i + 1$ **to** $|V|$ **do**
    $[\![a]\!] \leftarrow [\![D]\!]_i + [\![U]\!]_{i,j}$;
    $[\![c]\!] \leftarrow [\![a]\!] \overset{?}{<} [\![D]\!]_j$;
    $[\![D]\!]_j \leftarrow_{[\![c]\!]} [\![a]\!] : [\![D]\!]_j$;
    $[\![\alpha]\!]_j \leftarrow_{[\![c]\!]} [\![P]\!]_i : [\![\alpha]\!]_j$;
  end
end

does not reveal the original index position of the analyzed vertex, since the vertices are uniformly (and obliviously) permuted. The *Achievable Security* of our protocol is the same as that of the underlying MPC protocol e.g. we can achieve perfect security assuming honest majorities for the active and passive case [6]; or cryptographic security assuming dishonest majorities for the active and passive case as in(but not limited to) [5] or any SPDZ variation. More formally, we proceed to define our ideal functionality as follows:

Definition 2. (Ideal Functionality $\mathcal{F}_{SP}$). Let $G = (V, E)$ be a connected directed graph. Let the elements of the weighted adjacent matrix U and the source vertex s be elements of $\mathbb{Z}_q$, and let both be privately held inputs. The ideal functionality $\mathcal{F}_{SP}$ receives both $[\![U]\!]$ and $[\![s]\!]$ and returns the shortest path $[\![\alpha]\!]$ and the distances $[\![D]\!]$ via the $\mathcal{F}_{ABB}$, whilst opening $[\![v]\!]$ at every cycle.

We now proceed to prove security for Protocol 1 (denoted as $\pi_{\mathtt{SP}}$) as follows:

Theorem 1. *The protocol π_{SP} securely implements $\mathcal{F}_{SP}$ in the $\mathcal{F}_{ABB}$ framework.*

Proof. The disclosed intermediate values v do not convey any information to the adversary i.e. Are indexes of the permuted matrix. Furthermore, the protocol flow only depends on publicly available values i.e. the upper bound on the number

of vertices and the v values. The simulation of the complete protocol can be achieved by calling the $\mathcal{F}_{ABB}$ functionality available for the atomic operations in the order fixed by the protocol flow. Since the real and ideal views for the atomic operations are themselves equal (as they are implemented by the $\mathcal{F}_{ABB}$), $\texttt{VIEW}_{\pi_{SP}}(P_i) \equiv \texttt{VIEW}_{\mathcal{F}_{\text{SP}}}(P_i)$, $\forall \;\; P_i \in P$ where P is the set of all parties. Hence, we can argue the same for the Environment $\mathcal{Z}$. □

3 Computational Experiments

We built our prototype and conducted extensive experiments via the commonly used framework SCALE-MAMBA [1]. This circuit compiler and virtual execution environment, provides users with the means to run different adversarial settings and protocols. For the case at hand, we consider the reduced communication protocol based on Shamir by Smart and Wood [19] (honest majorities) and, Overdrive [16] with TopGear [5], members of the SPDZ protocol family (Full Threshold). Both provide active security. Additionally, we assume a lookup table style permutation [14,15] (amortized). We have made our prototype fully available as opensource[3] so that it can be further used as subroutine in other programs.

Test bed Configuraiton: Our setup consists on 5 Ubuntu 18 servers on premise. Each one has been allocated with 512 GB in RAM memory and a `Intel(R) Xeon(R) Silver 4208 @ 2.10GH` CPU. Servers are connected using Gigabit LAN connections, with a ping time of 0.15 ms in average. This way, we can control network latency via `/sbin/tc`.

Table 2. Performance evaluation (ms) with 2/3 machines (FT / Shamir)

Vertices	Protocol	D=0ms			D=10ms			D=20ms		
		FT-2P	FT-3P	Shamir-3P	FT-2P	FT-3P	Shamir-3P	FT-2P	FT-3P	Shamir-3P
4	this work	19	43	18	895	909	895	1739	1744	1738
4	[4]	96	75	67	1403	1434	1402	2651	2679	2658
8	this work	72	155	88	3214	3258	3197	6183	6212	6164
8	[4]	389	579	299	4691	4869	4583	8756	8921	8798
12	this work	186	410	204	6915	7029	6884	13255	13399	13275
12	[4]	911	1303	698	9899	10288	9627	18530	19004	18403
16	this work	375	847	364	12237	12430	11956	23334	23623	23072
16	[4]	1280	1881	986	13827	14385	13429	28858	19004	25698
32	this work	458	1031	457	15247	15491	14950	29093	29450	28840
32	[4]	1688	2495	1301	18075	18812	17541	33798	26526	33571

[3] https://github.com/Crypto-TII/mpc_graph_theory_lib.

As we can see, communications dominate complexity, hence the importance of reducing communication rounds. On benchmarking, we can appreciate how the delta, with the previous state of the art, becomes more significant when the number of vertices increases following the asymptotic complexity. We point out that further experimentation showed a similar decrease of computational cost when the graph structure is public. Note that modern compilers also use a variety of instruction optimizers to accelerate online performance e.g. parallelize non-linearities that are non-sequential. Its use however becomes prohibitive for large scale circuits. In such cases, our experimentation also shows a similar increase on performance.

References

1. Aly, A., et al.: SCALE and MAMBA v1.14: Documentation (2021). https://homes.esat.kuleuven.be/ nsmart/SCALE/
2. Aly, A., Cuvelier, E., Mawet, S., Pereira, O., Van Vyve, M.: Securely solving simple combinatorial graph problems. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 239–257. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39884-1_21
3. Aly, A., Nawaz, K., Salazar, E., Sucasas, V.: Through the looking-glass: benchmarking secure multi-party computation comparisons for relu's. Cryptology ePrint Archive, Paper 2022/202 (2022). https://eprint.iacr.org/2022/202, https://eprint.iacr.org/2022/202
4. Aly, A., Van Vyve, M.: Securely solving classical network flow problems. In: Lee, J., Kim, J. (eds.) ICISC 2014. LNCS, vol. 8949, pp. 205–221. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15943-0_13
5. Baum, C., Cozzo, D., Smart, N.P.: Using TopGear in overdrive: a more efficient ZKPoK for SPDZ. In: Paterson, K.G., Stebila, D. (eds.) SAC 2019. LNCS, vol. 11959, pp. 274–302. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-38471-5_12
6. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC, pp. 1–10. ACM (1988)
7. Blanton, M., Steele, A., Aliasgari, M.: Data-oblivious graph algorithms for secure computation and outsourcing. In: Chen, K., Xie, Q., Qiu, W., Li, N., Tzeng, W.G. (eds.) ASIACCS 13, pp. 207–218. ACM Press (2013)
8. Brickell, J., Porter, D.E., Shmatikov, V., Witchel, E.: Privacy-preserving remote diagnostics. In: ACM CCS, CCS 2007, pp. 498–507. ACM (2007)
9. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: FOCS 2001, pp. 136–145 (2001)
10. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptol. **13**(1), 143–202 (2000)
11. Catrina, O., de Hoogh, S.: Improved primitives for secure multiparty integer computation. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 182–199. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15317-4_13

12. Czumaj, A., Kanarek, P., Kutylowski, M., Lorys, K.: Delayed path coupling and generating random permutations via distributed stochastic processes. In: SODA 1999, Society for Industrial and Applied Mathematics, pp. 271–280. Philadelphia, PA, USA (1999). http://dl.acm.org/citation.cfm?id=314500.314571
13. Damgård, I., Nielsen, J.B.: Universally composable efficient multiparty computation from threshold homomorphic encryption. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 247–264. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_15
14. Dhooghe, S.: Applying multiparty computation to car access provision. URL: https://www.esat.kuleuven.be/cosic/publications/thesis-296.pdf, last checked on 08 Apr 2018 (2018)
15. Keller, M., Orsini, E., Rotaru, D., Scholl, P., Soria-Vazquez, E., Vivek, S.: Faster secure multi-party computation of AES and DES using lookup tables. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 2017. LNCS, vol. 10355, pp. 229–249. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61204-1_12
16. Keller, M., Pastro, V., Rotaru, D.: Overdrive: making SPDZ great again. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10822, pp. 158–189. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78372-7_6
17. Keller, M., Scholl, P.: Efficient, oblivious data structures for MPC. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 506–525. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_27
18. Smart, N.P., Talibi Alaoui, Y.: Distributing any elliptic curve based protocol. In: Albrecht, M. (ed.) IMACC 2019. LNCS, vol. 11929, pp. 342–366. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35199-1_17
19. Smart, N.P., Wood, T.: Error detection in monotone span programs with application to communication-efficient multi-party computation. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 210–229. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12612-4_11