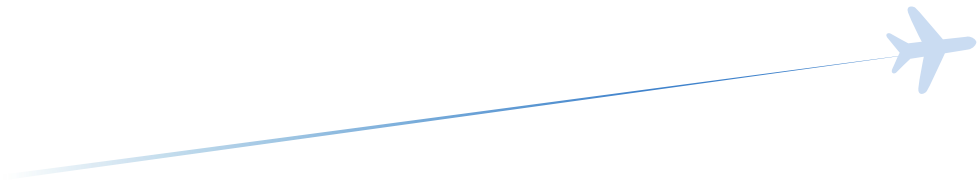

D3.1 Report on State-of-the-Art of Relevant Concepts

Deliverable ID:	**D3.1**
Dissemination Level:	**PU**
Project Acronym:	**SlotMachine**
Grant:	**890456**
Call:	**H2020-SESAR-2019-2**
Topic:	**SESAR-ER4-27-2019 Future ATM Architecture**
Consortium Coordinator:	**Frequentis**
Edition Date:	**28 July 2021**
Edition:	**01.01.01**
Template Edition:	**02.00.02**

Founding Members
EUROPEAN UNION EUROCONTROL
SESAR JOINT UNDERTAKING

Authoring & Approval

Authors of the document

Name/Beneficiary	Position/Title	Date
Thomas Loruenser / AIT	Senior Scientist	2021-07-01
Florian Wohner / AIT	Research Engineer	2021-07-08
Roman Karl / AIT	Scientist	2021-07-01

Reviewers internal to the project

Name/Beneficiary	Position/Title	Date
Christoph Fabianek / Frequentis	Technical Manager	2021-07-12
Eduard Gringinger / Frequentis	Project Manager	2021-07-19

Approved for submission to the SJU By - Representatives of beneficiaries involved in the project

Name/Beneficiary	Position/Title	Date
Christoph Fabianek / Frequentis	Technical Manager	2021-07-20
Eduard Gringinger / Frequentis	Project Manager	2021-07-20
Marie Carré / SWISS	WP 5 Leader	2021-07-23 (silent approval)
Nadine Pilon / EUROCONTROL	WP2 Co-Leader	2021-07-23 (silent approval)
Christoph Schuetz / JKU	WP2, 3 Leader	2021-07-23

Rejected By - Representatives of beneficiaries involved in the project

Name/Beneficiary	Position/Title	Date
-	-	-

Document History

Edition	Date	Status	Author	Justification
01.00.00	2021-05-10	First draft	T. Loruenser	Initial structure
01.00.01	2021-07-01	Internal release	T. Loruenser	Review process
01.01.01	2021-07-28	Final release	T. Loruenser	Ready for delivery

SlotMachine

A PRIVACY-PRESERVING MARKETPLACE FOR SLOT MANAGEMENT

This deliverable is part of a project that has received funding from the SESAR Joint Undertaking under grant agreement No 890456 under European Union's Horizon 2020 research and innovation programme.

Abstract

This document describes the state-of-the-art in technical areas relevant for SlotMachine. In particular it gives an overview on privacy enhancing technologies which are used to establish privacy preserving marketplaces in a decentralised fashion. It further analyses the role of blockchain in this context and how it could be leverages to further increase security and trust in the system. For all key technologies used we did a literature review and analysed existing open source software frameworks for their maturity and performance.

One of the core technologies analysed is multiparty computation (MPC) which enables the evaluation of functions while keeping the inputs private. It will be used in SlotMachine to replace the trusted party typically needed to evaluate the private bids and priorities of airspace users during the optimization step. Secondly, we looked into methods for verifiable computing and how the overall process could be made more transparent although the private inputs must be kept confidential. Efficient and practical zero-knowledge proof systems have been studied and the most promising candidates to realize some form of public verifiability and traceability are highlighted. Finally, blockchain solutions to realize a dedicated permissioned ledger as a trust anchor in SlotMachine have been analysed. The scalability and performance were measured in practical deployment scenarios.

Based on the results from our research and testing we finally conclude the results, give some recommendations for further development and propose a first high-level architecture combining the technologies.

Table of Contents

1 Introduction 6

1.1 Purpose of the document 6

1.2 Scope 6

1.3 Intended readership 6

1.4 Structure of the document and relation to other deliverables 6

2 Privacy Preserving Market Places 8

2.1 Overview 8

2.2 Market Places and Privacy 9

2.3 Secure Markets on Blockchains 11

2.4 MPC Supported Markets (on Blockchain) 13

2.5 Privacy Aspects of Blockchain Tokens 14

3 Multiparty Computation (MPC) 16

3.1 Overview 16

3.2 Relation to SlotMachine Requirements 17

3.3 Overview of MPC Frameworks 18

3.3.1 MP-SPDZ (forked from SCALE-MAMBA) 18

3.3.2 MPyC 19

3.3.3 FRESCO 19

3.3.4 ObliVM 20

3.3.5 Obliv-C 20

3.3.6 ABY 21

3.3.7 EMP-SH2PC 21

3.4 Evaluation Results 22

3.4.1 Systems based on Garbled Circuits 24

4 Verifiable Computing (VC) for Auditing 25

4.1 Introduction 25

4.1 Relation to SlotMachine Requirements 26

4.2 Overview of Protocol Families 27

4.2.1 Systems based on linear PCPs 27

4.2.2 Discrete log-based systems 27

4.2.3 Short PCP 28

4.2.4 Other Solutions 29

4.3 Overview of Available Frameworks 29

4.3.1 PySNARK 29

4.3.2 libsnark 30

4.3.3 libSTARK 31

4.3.4 Bulletproofs 31

4.4 Evaluation Results....32
5 Evaluation of Blockchains.... 33
5.1 Introduction....33
5.2 Tendermint evaluation....33
6 Conclusions and Recommendations.... 36
7 References.... 38

List of Tables

Table 1 MP-SPDZ speed of basic operations on vectors....22
Table 2 MPYC measurement results for basic functionality....23
Table 3: Peformance comparison of GC frameworks....24
Table 4: Performance figures for proof systems....33

List of Figures

Figure 1: Comparison of variants based on garbled circuits (GC)....25
Figure 2: The block interval time for the first experiment....34
Figure 3: The block interval time for the second experiment....35
Figure 4: The block interval time for the third experiment....35
Figure 5: Proposal for the cryptographic....37

1 Introduction

1.1 Purpose of the document

This document summarizes the state-of-the-art for relevant concepts, methods, technologies and frameworks which are used in SlotMachine to achieve security, privacy and transparency. In particular, cryptographic techniques from the fields of multiparty computation (MPC), verifiable computation (VC) and blockchain have been researched and evaluated. Besides the literature review, we also report results from performance and scalability testing done with the most promising open-source frameworks. The tests were done to learn more about practical aspects of available solutions and their shortcoming when it comes to adoption for SlotMachine. The document basically reports results from the work conducted in T3.1 (MPC), T3.2 (blockchain) and T3.3 (VC) and led to recommendations for the system design and requirements as well as a R&D agenda for the subsequent tasks.

1.2 Scope

The document covers the state-of-the-art for security, privacy and transparency aspects in SlotMachine. It is focused on technical means and cryptographic solutions which will be potentially researched and used in the project. The scope is on the Privacy Engine component as defined in D2.2, which comprises most of the cryptographic solutions. Additionally, blockchain is foreseen to be used as a ledger by all participants and to also run the token system.

1.3 Intended readership

This document is intended for both internal and external audiences. Internally it is mainly aimed at the technical team members in WP3 but also contributes important insights for work in WP2 and WP4. Additionally, it serves as the basis for further selection of technologies and existing implementations in subsequent WP3 developments. In particular, the insights are key for the design of the Privacy Engine (PE) and the integration of blockchain in SlotMachine, especially with respect to security, privacy and trustworthiness. However, it could be also a useful resource for other project participants and the public because it provides a comprehensive overview of novel cryptographic methods not widely known to non-cryptographers and security experts.

1.4 Structure of the document and relation to other deliverables

The remainder of the document is structured along the main technologies analysed and comprises the following chapters.

Chapter 2 gives an overview on privacy preserving marketplaces and how they could be realized with the technologies proposed for SlotMachine.

Chapter 3 presents relevant aspects of multiparty computation (MPC) which enables the evaluation of functions while keeping the inputs private. Most important open source frameworks are compared and benchmarking results presented to give a first indication for MPC usage in SlotMachine.

Chapter 4 is focused on methods for verifiable computing (VC) and how the overall process could be made more transparent although the private inputs must be kept confidential. Efficient and practical zero-knowledge proof systems have been studied and the most promising candidates to realize some form of public verifiability and traceability are highlighted.

Additionally, in chapter 5 blockchain solutions to realize a dedicated permissioned ledger as a trust anchor in SlotMachine have been analysed. The scalability and performance were measured in practical deployment scenarios.

Based on the results from our research and testing we finally conclude in chapter 6, give some recommendations for further development and propose a first high-level architecture combining the technologies.

2 Privacy Preserving Market Places

2.1 Overview

SlotMachine is dedicated to the development of a platform to optimize management of airport departure and landing slots. The platform shall enable a more flexible, fast and scalable semi-automated flight prioritisation process for airlines in a fair and trustworthy way. Built with a privacy-first approach it will protect sensitive airline data from competitors and airport operators but fully unleash the potential of inter-airline flight swapping in real-time. The goal is to minimise the overall costs caused by delays, which can be done by allowing airlines to almost dynamically rearrange and prioritise certain flights. This is already possible within a fleet [1] but to minimise costs, airlines need to be able to prioritise delayed flights across airline boundaries and would like to do so without prolonged negotiations. This is inherently difficult because airlines as competitors are very careful not to disclose any business secrets such as the flight-specific estimated costs associated with delays of different severities.

SlotMachine tackles this challenge by combining tools for privacy-preserving computation on data based on multiparty computation (MPC) with evolutionary algorithms and blockchain technology to build a decentralised system that enables collaboration for optimal flight sequencing in challenging conditions. It introduces a new approach to cooperative slot management and establishes a platform for on-demand automated operation. The platform serves as a marketplace for airlines with the overall aim of developing a novel flight prioritisation platform – the SlotMachine architecture – to improve the use of available resources at airports and reduce costs for airlines.

The market mechanism as currently foreseen in the project resembles a mixture of optimization problem and auctions system. It is at its core an optimization task closely related to queue/slot reordering/scheduling but also takes into account preferences of airspace users (AUs) and enable them to bid on certain flights in order to prioritize them. The bidding will be based on a dedicated token or credit system which is used to establish fairness and equity in the long run, i.e., over multiple reordering sessions. However, it should be stressed that the credits are not intended to be traded in a classical fashion with fiat currencies on other markets, they should only help in supporting fairness and equity between airlines.

In this report we review the state-of-the-art of relevant core technologies to support the increased privacy and trustworthiness requirements envisioned by the project. From an architectural perspective SlotMachine will be a decentralized platform where no single entity has full control over all information and decisions. The basic architectural concept in SlotMachine from a technical point of view is presented in D2.2 and the requirements are assessed in D2.1, including the security, privacy and transparency requirements mainly supported by the technologies analysed here.

SlotMachine makes heavy use of modern cryptography and the main cryptographic components will be combined into the Privacy Engine (PE) which encapsulates all complex cryptographic tasks in an easy to use manner from the rest of the platform and represents the (distributed) place where sensitive information is managed, i.e., specifically confidentiality is protected. In essence, the Privacy Engine is a module enabling multiparty computation which processes sensitive information in encrypted form only. If encrypted information is never restored for processing —as is typically the case in conventional cloud computing— security and privacy gains can be realized. This is especially true in

outsourcing scenarios where multiple stakeholders share private information to collaborate on joint data sets. However, because computation on encrypted data introduces significant computation and communication overhead we had to carefully analyse and assess existing approaches to understand their properties and limitations. We also report results of extensive performance testing and code analysis of available open-source frameworks for MPC to compare their capabilities and maturity with respect to our application.

Besides the protection of sensitive AU data, the system as a whole should also be trustworthy. It would be especially interesting to add technical means for auditing and other assurance mechanisms in order to convince users of correct consideration of their input data in individual optimization runs and fair treatment over multiple runs. Blockchain is adequate for this task and will be used to maintain metadata about optimization runs. The main property of blockchain relevant for our use case is immutability, i.e., it serves as a public append-only log to store information about flight prioritization in the past, and optionally to also run the credit system envisaged. The blockchain itself should be run as distributed as possible but still in a closed user group, the AUs. In this report we will therefore look particularly into permissioned blockchain/consensus protocols and their scalability.

Additionally, to combine both technologies — MPC and blockchain — in a fruitful way we had to investigate possibilities to store data into blockchain in oblivious form while still making sense of the data. In particular, we looked into possibilities to use zero-knowledge proof techniques to maintain confidentiality of sensitive data but still enable public verifiability aspects, i.e. let AUs check the validity of flight swaps but without leaking prioritization decisions. Therefore, we analysed existing methods and open source frameworks, which also have been tested.

2.2 Market Places and Privacy

Although a heuristic optimization algorithm is used to compute swapping proposals, the idea of user-driven prioritisation is also closely related to auction systems, especially with respect to prioritization (bidding) of individual preferences. Therefore, we briefly mention most important auction mechanisms and important privacy aspects regarding different phases. This is particularly interesting because privacy aspects are key in auctions to work correctly and achieving fair market prices. If individual bidders gain additional information they are not supposed to know, they have a clear advantage and could significantly impact the markets. Especially, if trusted authorities are needed to run auctions, they could become a single point of trust and failure. Therefore, in many situations —as for flight swapping in SlotMachine— it is not feasible to find an entity which is fully trusted by all participants to handle sensitive information and to correctly process them without bias. Imagine a basic sealed-bid auction where the trusted authority colludes with a bidder which can then easily win any auction without paying the true price, but just a bit more than the second highest bidder. Therefore, we are interested in decentralized sealed-bid type of auctions/prioritization without any trusted auctioneer. This is particularly interesting because the first practical applications for MPC were indeed auctions [2] with exactly that goal.

As summarized in [3] there are four main types of auctions which are of practical interest in typical real-live scenarios:

1. First-price sealed-bid auctions (FPSBA). Bidders submit their bids in sealed envelopes and hand them to the auctioneer. Subsequently, the auctioneer opens the envelopes to determine the bidder with the highest bid.

2. Second-price sealed-bid auctions (Vickrey auctions). It is similar to FPSBA with the exception that the winner pays the second highest bid instead.
3. Open ascending-bid auctions (English auctions). Bidders increasingly submit higher bids and stop bidding when they are not willing to pay more than the current highest bid.
4. Open descending-bid auctions (Dutch auctions). Auctioneer initially sets a high price, which is gradually decreased until a bidder decides to pay at the current price.

Additionally, besides the standard auctions there are many special kinds of auction. Interestingly the first proposal for so called combinatorial auctions were based on slot-trading scenarios for airlines [4], [5]. This kind of auctions is fundamentally different from the main four types and imposes completely different challenges in the computation of winning bids. From a computational perspective they are more like optimization problems than sorting tasks and can require a significant amount of computational work. However, they are based on the idea of bidding on bundles of resources instead of individual ones. In [4] they propose a market mechanism to trade starting and landing slots on different airports as bundles and therefore optimize more globally. In SlotMachine the focus is on already scheduled flight sequences for departures at a given airport, which is not directly related to the problem of combinatorial auctions, but it somehow also resembles an optimization problem.

Regarding security and privacy of auctions, different goals could be desirable. Informally we can distinguish the following properties from secure auctions, as have been defined in [5]:

1. Bid privacy. All bidders cannot know the bids submitted by the others before committing to their own. This property is also guaranteed even in a collusion with a malicious auctioneer.
2. Posterior privacy. Given a semi-honest auctioneer, all committed bids are maintained private from the bidders and public users.
3. Bid Binding. Once the bid interval is closed, bidders cannot change their commitments.
4. Public verifiable correctness. The auction contract verifies the correctness of the auctioneer's work to determine the auctioneer winner.
5. Financial fairness. Bidders or auctioneer may attempt to deviate from the protocol and prematurely abort to affect the behaviour of the auction protocol. The aborting parties are financially penalized while honest parties are refunded after a specific timeout.
6. Non-Interactivity. Bidders do not participate in complex interactions with the underlying protocol of the auction contract. In fact, no extra communications between the bidders and the auction contract are required aside from the submission of the bid commitments and the associated opening values.

In SlotMachine we are clearly focusing on bid privacy (1.) in the sense that information to prioritize certain flight by airlines must be kept private. At the time of writing, in SlotMachine flight prioritization is intended by means of a weight map and additional credits which can be used to further prioritize certain flights. This means that exactly this weight map as well as the additional credits spent by AUs on their flights have to stay encrypted and are only allowed to stay within the privacy engine. Furthermore, we also do not want to reveal the private information provided by airlines after the optimization run, thus, we are also aiming for posterior privacy (2.). Bids in the sense of SlotMachine contain many business secrets about internal cost structures of airlines and contain a lot of information also about related flights and recurrent situations. Keeping AUs preferences secret over the full life cycle in the system is essential to keep the airlines participating. Also bid binding (3.) is very important

in SlotMachine and will be supported, also by the help of blockchain. Finally, also public verifiability (4.) is aspired in SlotMachine, because this is seen as an important property to increase the trust in the platform. This goal is especially challenging when it has to be combined with posterior privacy. If all bids are revealed after the new swapping proposal was accepted, things would be easier, because the original bids could just be opened and verified. However, to do it in encrypted form modern cryptographic techniques are needed in form of "Zero-Knowledge Succinct Non-Interactive Argument of Knowledge" (zk-SNARKS), especially ones which can be combined with MPC. Finally, financial fairness (5.) and non-interactivity (6.) are not in the focus of SlotMachine, although we have planned to look into misuse cases regarding AU and also foresee some public verifiable means to check the correctness of AU input data. Nevertheless, because all parties are known to each other and because we expect the AUs to participate on a continuous basis, it would be rational for them to play with the rules to be not expunged from the system.

2.3 Secure Markets on Blockchains

Since the emergence of Bitcoin and Ethereum, there has been a huge hype around blockchains. Although the hype may be not completely justified from a technical perspective, these novel systems still have some potential in several application areas. A system that is built around a blockchain as main data structure mainly consists of three parts:

1. a mechanism for processing transactions
2. a consensus mechanism
3. a network protocol

The network protocol is an important aspect, but it is the least novel of these three parts and does not differ much from other distributed systems. The main innovative feature of blockchain-based systems is often considered to be the consensus algorithm. Indeed, Bitcoin's proof-of-work drew a lot of attention from the scientific community. On the other hands, it cannot be denied that this kind of consensus mechanism suffers from huge problems when applied in practice, the main problem being the vast energy consumption. There exist several concepts to alleviate this point, for example proof-of-stake, but also mechanisms that deviate from the Nakamoto consensus. But the part that offers the most interesting possibilities for future systems in a wide range of different application areas is the way how transactions are encoded, interpreted and processed. The transactions that are stored in the blockchain are connected by a logic that is formulated in some less or more expressive programming language. Furthermore, several cryptographic components are included to reduce the minimal level of trust that is expected of a single user in these multi-user environments. The next two subsections explain the components that can be employed when building markets by utilising blockchain-based systems.

Specific markets often follow their own set of rules, protocols and processes, which can be supported by code that allows a level of automation. Because markets are distributed by nature and feature different sets of actors, this could be a field of research with a lot of potential in the context of blockchain-based systems. Patterns that occur often in various markets are schemes that resemble auctions or have similar properties than auctions. Indeed, auctions on smart contracts have drawn a lot of attention in research and are also one of the basic examples that are brought up often to show the potential of programmable blockchain environments.

One type of markets that is especially relevant when examining the question how and if the application of blockchains can be beneficial in future are markets for electrical energy. One reason for this are the necessary changes due to the need of including more renewable energy sources into the grids and thereby changing a lot of requirements for them. Energy grids are expected to get a lot more distributed in comparison to the relatively centralised approach of current grids. Also in this context, prices can be regulated by auctions. Ul Hassan et al. [6] presented an approach for auctions on microgrids that also takes several privacy considerations into account.

Another relevant area is the allocation of resources of some kind of infrastructure. Jiao et al. [7] investigated the topic of auctions for computing resource allocation for cloud computing. They used a blockchain-based system with proof-of-work as consensus mechanism. A different approach that focusses on efficiency to be suitable for mobile environments is presented in [8].

Similar to the classic (forward) auctions are reverse auctions where the roles of buyer and seller are swapped. This setting is relevant for construction or infrastructure orders with a trusted actor, possibly represented by a state, with the central role of a buyer. Franco et al. [9] described a reverse auction for infrastructure supply, including network function virtualization in their design.

Throughout the several application areas of auctions some requirements and properties pop up regularly and can be viewed upon with a more general perspective independently of the specific area. Many auctions do not rely solely on public information but require some level of privacy. It is not obvious how to get a good level of privacy on an inherently distributed environment with a blockchain that should be readable by all the participants of an auction to establish a layer of trust. Galal et al. [10] presented a protocol for auctions in an Ethereum network. It included sealed bids with a public key encryption and the usage of zero-knowledge proofs. Although strong privacy properties for an auction protocol are in general achievable, they are often not desirable in practice due to their strong impact on the performance. An auction scheme with a good trade-off between privacy properties and performance was given in [11]. They focussed on reducing the number of blocks necessary to complete the auction. This is an important performance aspect on a blockchain-based system because the transaction throughput is often limited as is the rate at which new blocks are generated.

As with the general spending of tokens, there can be a connection to not only virtual, but physical objects in auctions. Indeed, most of the more interesting application areas deal with some kind of physical objects. With smart contracts, this is often done by introducing oracles. In this context, an oracle is a trusted entity that reports the relevant part of the state outside of the blockchain system on the blockchain. Special measures have to be taken to ensure the trustworthiness of this approach as the oracle has such a critical role. Omar et al. [12] presented a concept for auctions with trusted timer oracles on Ethereum.

2.4 MPC Supported Markets (on Blockchain)

MPC can be considered the most practical approach for generic computation on encrypted data. This means, that virtually any function can be computed in an MPC system in principle, however, due to the overhead introduced by MPC protocols they are by orders of magnitudes slower than a classical computer.

Nevertheless, the first realization of a real application was demonstrated in an auction held for the Danish sugar beets market in 2009 [2]. This was the first large-scale and practical application of multiparty computation and enabled farmers to get a fair market clearing price. They used a local setup with three computers in the same room and ran a semi-honest MPC protocol to calculate the clearing price. About 4000 values for prices were supported at most and 1229 bidders participated in the auction. The inputs from the individual bidders were encoded by verifiable secret sharing and the computation lasted for half an hour.

The basic problem of this setting is the lack of scalability of the MPC protocols itself. This resulted in a setup with 3 nodes which prevents from the clients to directly participate in the computation but encode the inputs, which still leads to a form of outsourced computation, although a distributed one. The improved security in this setting is evident, but to further increase the trustworthiness of the system some form of public verifiability would be desirable.

To cope with this issue new research combined the mechanisms with blockchain and zero-knowledge proof techniques. The blockchain is the ideal candidate to be used for storage of relevant audit data in an accessible manner, however, because all data written to the blockchain is visible to every party additional machinery is required. ZKP protocols enable parties to publish proofs about statements without revealing secrets per se (witnesses) and are therefore an ideal tool to integrate blockchain with the confidentiality preserving MPC functionality.

Sánchez [13] proposed Raziel, a system that combines MPC and ZKP to guarantee the privacy, correctness and verifiability of smart contracts. The idea underlying Raziel is a smart contract which also guarantees correctness of auctions besides the standard properties by leveraging the ZKP mechanism. The validity proofs can also be shown to third parties and are therefore publicly verifiable. Another approach to verifiable auctions has been presented in [3] and a software prototype can be found on GitHub[1]. The solution combined homomorphic commitments and ZKP together with a verifiable comparison protocol to achieve a secure FPSBA. The system is verifiable and privacy preserving against outsiders, however, a trusted auctioneer is still required because he learns all bids.

Furthermore, Blass and Kerschbaum [11] presented Strain, a protocol to implement sealed-bid auctions on top of blockchains that protects the bid privacy against fully malicious parties. The protocol basically stores encrypted bids in the blockchain. By using a specific encryption scheme with homomorphic properties in combination with bitwise encryption, they enable bidders to run interactive protocols in zero knowledge generating relations proofs and therefore, support auctions in a peer to peer fashion. Albeit being scalable by the peer-to-peer nature, the protocol still needs a semi-

[1] https://github.com/HSG88/AuctionContract

trusted auctioneer as arbiter and requires all parties to be online during all auction phases. It also leaks the full order of all bids compared to the winning bid as required by auctions.

Kosba et al. [14] presented Hawk, a framework to establish privacy preserving smart contracts on the Ethereum blockchain. Hawk is intended to protect transaction data on chain by leveraging zero-knowledge proof techniques. The goal was towards easy to use framework providing a compiler managing the cryptographic tasks. This work could be relevant for the specific credit system embedded in SlotMachine which also has some privacy requirements. Up to our knowledge, the Hawk framework has still not been released yet on the project homepage[2].

In another work, Galal and Youssef [15] utilized Zero-Knowledge Succinct Noninteractive Argument of Knowledge (zk-SNARK) [16] to realize privacy friendly auctions on a blockchain. The solution is not well suited because it makes use of a trusted auctioneer who learns the bids. This is contrary to our goals; however, the approach contains interesting aspects and by realizing the auctioneer in a distributed fashion by MPC, the system closely resembles the data flow in SlotMachine.

Additionally, cryptocurrencies have been used to incentivize fairness and correctness, and avoid deviations from the MPC or ZKP protocol. In this systems money has to be escrowed in deposits which are only returned if the behave honestly. This in effect encourages parties to strictly follow the protocols to avoid the financial penalty. Protocols in this direction have been proposed in [17]–[20].

2.5 Privacy Aspects of Blockchain Tokens

The usage of tokens is still the only widely applied use case for blockchain-based systems. Tokens can be used as a kind of money or currency, either with a concrete value in relation to legal currencies or with a more abstract value. Such tokens are called fungible or interchangeable tokens. There are also non-fungible tokens, which represent some kind of collectible and have some properties that render each token potentially individual. A currency, thus a fungible token, is often build-in in a blockchain-based system, like the currency Bitcoin in the Bitcoin system and Ether in Ethereum. The programming capabilities allow to build further tokens on top of those systems. To that regard, Bitcoin‘s flexibility is more limited due to its programming language, which is not Turing-complete. Ethereum allows the creation of more complex programs, which are called smart contracts. Ethereum is also the platform with the broadest application of various tokens. In order to have some compatibility between the tokens and to allow the reuse of reviewed interfaces and code, there exist several token standards, which are described with Ethereum Improvement Proposals. The main standard for fungible tokens is ERC-20 [21]. For non-fungible tokens, Ethereum’s most widely used standard is ERC-721. There are a lot more properties which have to be taken into account when choosing or designing a suitable token for specific use cases. One extensive classification is presented by Oliveira et al. [22]. There are tokens with a connection to physical objects and purely virtual tokens. Furthermore, there are various more technical aspect like the use of layers beside the blockchain.

Bitcoin was designed with features which allow a level of privacy. Bitcoin addresses are not directly linked to an identity, which means that the identity of an account can be kept secret. This kind of anonymity is not very strong, because it is possible to extract information from transactions and their senders and receivers, and sometimes referred to as pseudo-anonymity [23]. Only if the behaviour of

[2] http://oblivm.com/hawk/download.html

a user is accordant with certain privacy guidelines, its anonymity can be kept. There exist also a range of services, like CoinJoin [24] and CryptoNote that assist in obfuscating coin flows and protecting identities. There are also some crypto currencies with stronger privacy features build-in, like Monero [25], Zcash and Dash. Monero uses ring signatures to obscure the sender, ring confidential transactions to hide the amount and stealth addresses to hide the receiver. Zcash supports transactions with zk-SNARKs out of the box, in contrast to other crypto currencies, where they have to build laboriously on top.

For several reasons, the frame conditions of token usage on smaller permissioned networks differ from a token usage on large public networks. The typical consensus mechanism for large public networks is still a Nakamoto consensus based on proof-of-work. It only makes limited sense to apply this consensus mechanism to systems with low computational power. Frameworks like Hyperledger Fabric and Tendermint [26] deviate from the Nakamoto consensus and use stricter types of consensus instead, because they were designed with a focus on potentially small permissioned networks. Both of those frameworks also have capabilities for writing Turing-complete smart contracts and thus for creating tokens similar as with Ethereum. But also on permissioned networks privacy can be an issue. Androulaki et al. [27] presented one approach of a privacy-preserving token mechanism on top of Hyperledger Fabric where the permissioned setting is leveraged to fabricate NIZK proofs.

3 Multiparty Computation (MPC)

3.1 Overview

Multiparty computation is a protocol between a number of players $P_1,\ldots,P_n$ who each initially hold inputs and we can then securely compute some function f on these inputs. This should hold, even if players exhibit some amount of adversarial behaviour. The goal can be accomplished by an interactive protocol π that the players execute. Intuitively, we want that executing π is equivalent to having a trusted party T that receives privately x_i from P_i, computes the function, and returns y_i to each P_i. With such a protocol we can — in principle — solve virtually any cryptographic protocol problem. The general theory of MPC was developed in the late 80s and overviews of the theoretical results known can be found in [28], [29].

Many different MPC protocols have been proposed in the past and they achieve very different properties and security guarantees. The minimal cryptographic properties of any MPC protocol are

- Correctness
- Input Privacy

Additional properties are often desired, such as

- Fairness
- Guaranteed output delivery

Besides the security properties, another way to categorize and assess MPC protocols by their adversary models. The most important considerations when it comes to adversaries are:

- Threshold security versus security against dishonest majority
- Unconditional versus computational security
- Active versus passive security
- Adaptive versus static security
- Synchronous versus asynchronous communication
- Broadcast channel versus point-to-point communication

From the very basic performance figures given by the nature of different solutions in SlotMachine we opted to start with the most efficient protocols for active security in the honest majority model. First tests already showed that dishonest majority protocols are performance wise out of any feasible range for the given requirements and do not fit to our needs. Therefore, in the following we will shortly mention the most efficient solutions published for this type of protocols which seem to integrable with existing open-source solutions and have optimization potential for batch processing.

Until recently protocols for active security have been considered rather inefficient. However, Lindell and Nof [30] showed more efficient protocols with great practical impact. They identified that privacy of active protocols is already preserved by existing passive protocols, and the protocols are susceptible to additive attacks. In [31] the protocols have been further improved and are the best known solution in the honest majority setting introduce only minimal overhead if processing can be batched. Furthermore, both protocols [30], [31] provide the highest level of security, i.e., information theoretic

security, and are therefore also quantum-safe. Additionally, [32] showed how to achieve fairness for the proposed protocols for active security with batch wise multiplication verification. Finally, [33] introduced a very efficient protocol to convert passive secure protocols into active secure ones with very little overhead. They basically apply a type of batch verification for multiplication operations, which can be done in bulk before the final result is revealed. Overall, the achievements in the last 3 years tremendously improved the state-of-the-art for active secure protocols in the honest majority setting and will be the starting point for further developments in SlotMachine.

3.2 Relation to SlotMachine Requirements

MPC has been selected as a core technology in PE for privacy protection of sensitive data and to enable collaboration on them in a secure way. On the system level the technology will be used to address the following non-functional requirements defined in D2.1:

priv_1, priv_2, priv_3, priv_4, priv_8, priv_9, priv_10	MPC will be used to fulfil the core confidentiality (privacy) requirements by computing on encrypted data
priv_6	MPC should assist in the detection of malicious clients although the input is encrypted

Additionally, in the selection process of MPC solutions and protocols following requirements are considered important when reviewing the state of the art:

perf_1, perf_2, perf_4	Main systems parameters to be supported by the PE for practical application of the technology
perf_9	The system should enable optimizations and interleaved/parallel execution
port_1	The system shall be based on open source solutions

Finally, because the PE is intended as an easy deployable and usable MPC system which is flexible but tailored to the specific tasks in SlotMachine, MPC will contribute to the following functional requirements. However, the protocols and frameworks analysed in this report are only matched against the basic functionalities of an MPC protocol and not application specific requirements which will be later integrated during the SlotMachine implementation phase.

Pe_2, Pe_7, Pe_8, Pe_9, Pe_13	Confidentiality requirements directly provided by the MPC protocol
Mpc_1, Mpc_9, Mpc_11, Mpc_15, Mpc_16, Mpc_17	Capabilities which have to be supported by the raw protocols (gates, circuits, data types, ….)

3.3 Overview of MPC Frameworks

For the analysis of existing open source software implementations of MPC protocols we started from the frameworks listed in AwesomeMPC-Frameworks[3] and selected the most promising candidates.

To evaluate the maturity and adequacy in SlotMachine different criteria were important for us:

1. Installation
2. Tests / Demos / Examples
3. Documentation
4. Programming Language

The amount of information and software prototypes seem overwhelming, especially if also the number of frameworks listed as retired[4] are considered. Out of the many available frameworks we identified MP-SPDZ and MPyC as the most relevant for SlotMachine and analysed them in detail. Additionally, we also investigated solutions with fixed numbers of parties (2 and 3). After initial review we focused on ObliVM, Obliv-C, ABY and EMP-SH2PC as most promising candidates.

3.3.1 MP-SPDZ (forked from SCALE-MAMBA)

Summary: Provides a wide variety of protocols for both arithmetic and boolean circuits; Supports many protocols, active and passive; implements a virtual machine that executes byte code compiled from a Python-like language; fork of SCALE-MAMBA (https://github.com/bristolcrypto/SPDZ-2); more flexible and extendible as SCALE-MAMBA; more protocols are supported and better for benchmarking than SCALE-MAMBA
Location: https://github.com/data61/MP-SPDZ
Authors / Maintainer: Marcel Keller
Dependencies / Required Software: • Make, GCC or Clang, Python, Libsodium, Boost, OpenSSL • MPIR with C++ support • NTL (optional)
Advantages: • Efficient and fast *Disadvantages:* • No obvious way to directly use from other software
Development Status: Very active, but only one developer (last commit July 2021)

[3] https://github.com/rdragos/awesome-mpc#frameworks
[4] https://github.com/rdragos/awesome-mpc#retired-software

3.3.2 MPyC

Summary: A generic framework that allows to write MPC-code directly in Python (forked from http://viff.dk/); based on secret sharing with support for honest majority multi-party protocol; secure against semi-honest adversaries
Location: https://github.com/lschoe/mpyc
Authors / Maintainer: Berry Schoenmakers
Dependencies / Required Software: • Python 3.6 or higher • gmpy2 (optional)
Advantages: • Easy to set up (there is a https://pypi.org/project/mpyc/ package) • Easy to integrate into existing software • Efficient and fast *Disadvantages:* • Slow on purely local computations
Development Status: Very active, but only one developer (last commit July 2021)

3.3.3 FRESCO

Summary: A Java framework implementing the SPDZ/SPDZ2k and TinyTables protocols
Location: https://github.com/aicis/fresco
Authors / Maintainer: The Alexandra Institute https://alexandra.dk
Dependencies / Required Software • Java [C/C++ (using JNI)]
Advantages: • Good documentation • Reasonably easy to use *Disadvantages:* • Limited protocols • SPDZ2k implementation is incomplete • Poor performance
Development Status: Somewhat active (last commit July 2021)

3.3.4 ObliVM

Summary: A Programming Framework for Secure Computation Compiler (written in Java) for a C-like language
Location: https://github.com/oblivm/ObliVMLang
Authors / Maintainer: Chang Liu, University of Maryland
Dependencies / Required Software • Java 8
Advantages: • None *Disadvantages:* • Could not be benchmarked properly because of non-deterministic program termination • Very inefficient (high number of gates, scaling badly) • Anecdotally, performance is bad and highly variable • Project has been abandoned
Development Status: Inactive (last commit November 2015)

3.3.5 Obliv-C

Summary: Compiler (written in OCaml) for the C-like Obliv-C language
Location: https://github.com/samee/obliv-c https://oblivc.org
Authors/Maintainer: Samee Zahur & David Evans, Security Research Group, University of Virginia
Dependencies / Required Software • libgcrypt • OCaml • OCaml libraries: batteries
Advantages: • Reasonably easy to install • Interfaces with C code • Best performance *Disadvantages:* • No active development in recent years • Sparse, incomplete documentation
Development Status: Somewhat active (last commit June 2021)

3.3.6 ABY

Summary: A Framework for Efficient Mixed-Protocol Secure Two-Party Computation C++ framework to programmatically build circuits from gates (supports AND, XOR, OR, ADD, MUL, SUB, GT, MUX, INV)
Location: https://github.com/encryptogroup/ABY
Authors / Maintainer: Daniel Demmler, Thomas Schneider and Michael Zohner, Cryptography and Privacy Engineering Group (ENCRYPTO), TU Darmstadt
Dependencies / Required Software • Boost • libgmp
Advantages: • Very efficient (low number of gates) • Good performance *Disadvantages:* • Could not be benchmarked properly because of non-deterministic program termination when run via script
Development Status: Somewhat active (last commit July 2021)

3.3.7 EMP-SH2PC

Summary: Semi-honest Two Party Computation Based on Garbled Circuits Part of the EMP (Efficient MultiParty computation) toolkit, a C++ framework to programmatically build circuits from functions
Location: https://github.com/emp-toolkit/emp-sh2pc
Authors / Maintainer: Xiao Wang, Alex J. Malozemoff and Jonathan Katz, University of Maryland
Dependencies / Required Software • other parts of EMP toolkit
Advantages: • *?* *Disadvantages:* • Little to no documentation
Development Status: Active (last commit July 2021)

3.4 Evaluation Results

Because the field of MPC implementations as a whole is still immature, it is difficult to judge the maturity of individual projects. This is reflected in the wide variety of approaches to interoperability. While some frameworks (like MP-SPDZ) provide only self-contained environments that exclusively read and write specially prepared files, others are almost regular libraries that integrate into their respective language environments – Python in the case of MPyC, Java in the case of FRESCO, and C in the case of Obliv-C.

Out of the above-listed frameworks, only MPyC and MP-SPDZ (both almost exclusively developed by one person) have seen sustained development over the respective project's lifetime. The others seem to be in maintenance mode. We therefore decided to investigate these two frameworks more thoroughly.

Table 1 MP-SPDZ speed of basic operations on vectors.

Delay [ms]	Max(5/100) [s]	Mul(100) [s]	Mul(1000) [s]
0	1,25967	0,97966	0,676558
10	21,3671	1,13602	10,6649
20	41,5928	2,21266	20,8486
30	61,6835	3,28712	31,0178
40	81,8286	4,36458	41,1688
50	101,969	5,43567	51,3525
60	122,148	6,51935	61,6066
70	142,311	7,59087	71,7877
80	162,559	8,66017	91,9108
90	182,768	9,73845	92,102
100	202,967	10,8134	102,274

We quickly found that it is important to compare computations under (somewhat) realistic conditions. A good example is given in Table **1** below, which compares the runtime of a simple multiplication of two vectors of 100 32bit integers each, and a slightly more complicated function that computes the maximum 5 elements out of a vector of 100 32bit integers, running on MP-SPDZ. Considered purely as a local computation, both functions are close to each other in their runtime, but as soon as any kind of network delay is added to the simulated runs (by way of the command `tc qdisc add dev lo root netem delay <N>ms;` the numbers given in the table refer to the two-way delay, so N is set to half the listed value), their runtimes differ from each other by about a factor 20. Notice this also

means that local performance (as measured for example on a single computer without any induced delay) does in no way indicate the performance to be expected under network conditions — in the one case, the runtime goes from one and a quarter second to more than twenty seconds, while in the other case, it goes from just under one second to slightly more than one second.

The picture is further complicated when comparing the results of MP-SPDZ with those of MPyC. While on a purely local level finding the maximum five elements out of a vector of 100 elements is more than ten times slower with MPyC, under any kind of realistic delay, MPyC is faster than MP-SPDZ. This is even more pronounced for the multiplication tests, which demonstrates another important factor in analyzing MPC frameworks: the necessity of domain-specific optimizations. MP-SPDZ uses an optimizing compiler, but of course this cannot guarantee that the resulting code is actually optimal. The multiplication tests for MP-SPDZ are written as one would write them for ordinary computations — first the two input vectors are multiplied element by element, then each element is revealed (to prevent the computation from being "optimized away" by the compiler). This means that for every vector element, two rounds of communication are necessary. Optimally, the multiplications would be batched such that the whole operation would take only one round of communication, but this would necessitate either a "sufficiently smart" compiler or the use of specialized vectorized operations by the programmer.

The MPyC version of the multiplication tests is written in the same style, but this framework does not use a compiler for a-priori optimization of computations, and instead relies on asynchronous computation and Python's highly optimized network stack, even though use of specialized vector primitives is also possible. See Table **2** for results.

Table 2 MPYC measurement results for basic functionality.

Delay [ms]	Max(5/100) [s]	Mul(100) [s]	Mul(1000) [s]
0	15,187632	0,195708	1,143771
10	25,560149	0,145964	1,266955
20	39,508425	0,262731	1,285819
30	54,102866	0,328410	1,259663
40	69,55985	0,393291	1,187533
50	84,931678	0,480012	1,145374
60	100,339924	0,498418	1,161129
70	115,254103	0,507936	1,356296
80	130,613401	0,749435	1,222092
90	145,464207	0,827996	1,255088
100	161,582399	0,704458	1,352546

In summary, estimating the performance achievable by MPC frameworks is difficult because it requires reasoning about "rounds of communication", which is a rather unintuitive notion that additionally depends on implementation details of the specific frameworks. However, we found MPyC to be the most appealing of the frameworks we investigated. It combines ease of setup and use with excellent efficiency and speed (if one keeps in mind that measurements obtained locally are in no way indicative of real-world performance). However, more testing is needed to better understand the performance for specific tasks encountered in SlotMachine and future work will do a deep dive in the implementation of optimization tasks for assignment problems with both, deterministic algorithms as well as evolutionary ones.

3.4.1 Systems based on Garbled Circuits

For evaluation, we implemented the same simple functionality in all garbled circuit frameworks presented in Section 3.2. The task was to find and return the maximum element of two unsorted lists, which is related to the main algorithms applied in basic auction protocols. In fact, the problem already comprises substantial complexity when it comes to the computational task involved. Every resulting program would be tested by running it in two different terminals on the same machine, being given two different, but equally long lists of integers.

Even though we were in all cases able to write programs that produced the correct solution, we found that not all of them would terminate reliably, that is: one or both of the running instances would keep running (sometimes consuming CPU time, sometimes not) and not shut down without user intervention. This prevented us from performing a systematic performance evaluation of several of the systems. The results are summarized in Table 3 and Figure 1.

input size	system	number of AND gates
2 * 100	ObliVM	602620
	ABY	12736
	OblivC (linear)	12800
	OblivC (recursive)	19104
2 * 500	ObliVM	121715434
	ABY	63936
	OblivC (linear)	64000
	OblivC (recursive)	95904

Table 3: Peformance comparison of GC frameworks

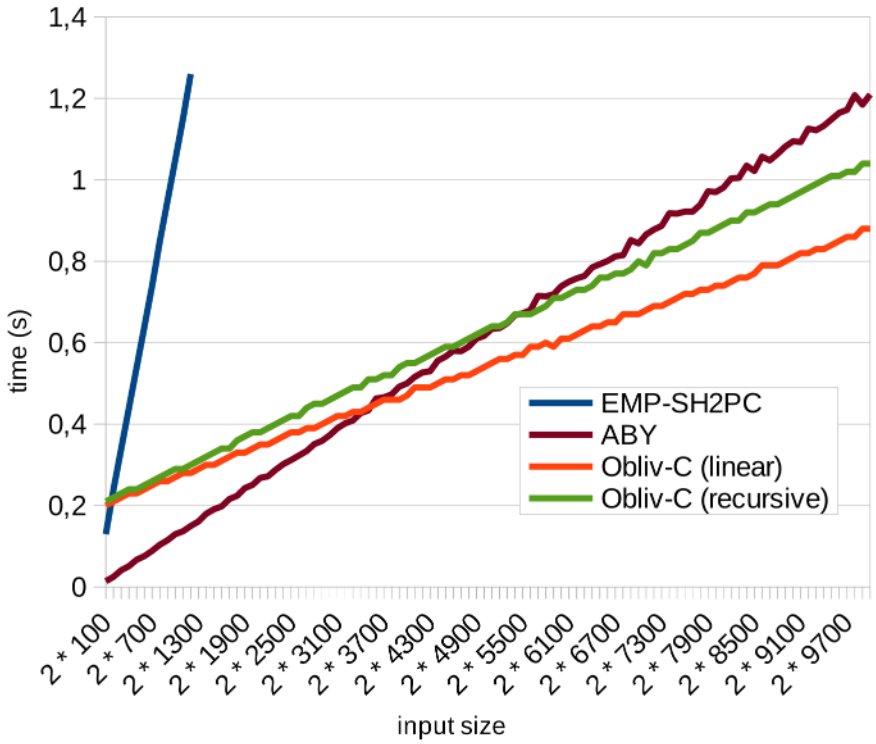

Figure 1: Comparison of variants based on garbled circuits (GC)

4 Verifiable Computing (VC) for Auditing

4.1 Introduction

Because MPC has only limited scalability and the feasible number of compute nodes will be between 3 and 7, the trustworthiness of the system could have some serious issues. In the end the system will still be an outsourcing scenario from the point of view of individual bidders who will not be able to run their own nodes and will have to trust the system. This means that security is governed by the non-collusion assumption of the MPC system which is reasonable for the privacy protection of individual bids, but not so appealing when it comes to transparency. With a system straightforwardly realized as described, a bidder has no means to check whether the computation was done correctly, and their own bid was also considered as expected. To allow bidders to do their own checks on the computation, we have to add additional technologies which introduce public verifiability for auctions allowing the bidder to check that the auction was run correctly, i.e., the highest bid won, and their own bid was also considered. This objective should hold up to the extreme case where all the parties involved in the computation are corrupted, and even if the party who wants to verify the result was not participating. However, the technologies have to work in a way that still protects the privacy of the bids, i.e., no single entity or dishonest minority is able to recover the bids.

A first mechanism for auditable MPC has been presented in [34] which was designed to make in particular the SPDZ type of protocols publicly auditable. The approach is already an improvement over

the traditional solution to this, which requires every party to commit to all their secret data and to prove in zero-knowledge the correct computation of every outgoing message they send. In principle, the approach introduces commitments on the input values which are then used as anchor to retrace the computation by the auditor or anyone who has access to the bulletin board. This approach introduces an auditor which has to perform substantial computational work. Furthermore, in addition to an already very expensive offline phase in the SPDZ protocol (which is already a bottleneck to be avoided in SlotMachine), the generation of proofs for all intermediate results leads to unusable performance for any practical application. Nevertheless, the idea of only proving relations on commitments instead of plaintext data is extremely useful for our purposes and the combination with blockchain.

Fortunately, zero-knowledge proof techniques made tremendous progress in recent years, especially since the introduction of “Zero-Knowledge Succinct Non-Interactive Argument of Knowledge.” These so called zk-SNARKS can be a game changer in the design of verifiable computing as desired in SlotMachine. Hence, it is now feasible to use zero-knowledge proof techniques to verify computation in a naive way and still get extremely efficient systems. In the following we give a short overview of the relevant protocols we identified which are also accompanied by open-source software implementations. As the development of the protocols itself would be out of scope for SlotMachine, we will research methods to integrate one of the best fitting existing solutions into our system and estimate the expected performance.

4.1 Relation to SlotMachine Requirements

VC has been selected as a method to increase the trust into the systems. It is complementary to MPC and will help to trace and monitor the correct working of the system for relevant stakeholder, ideally for all participants in the system. The technology will be used to address the following non-functional requirements defined in D2.1:

Perf_1, Perf_2, Perf_3	The system parameters must be supported to be practical.
Priv_6, Priv_7, Priv_12	Will be core method to verify the correct behaviour of AU and PE without revealing sensitive information. Cloud also help to protect privacy of credit system.

Because the VC sub-system must be compatible with the MPC technology used and integrated with PE, it is also relevant for the following functional requirements.

Pe_13, Pe_14, Pe_15, Pe_16, Pe_17	VC method must be interoperable with PE and also partly integrated with it.
Mpc_15, Mpc_16, Mpc_17	It will help to assure the correctness of the AU input in privacy preserving way.

4.2 Overview of Protocol Families

4.2.1 Systems based on linear PCPs

Recent advances in SNARKs (Succinct Non-interactive Arguments of Knowledge) are making it more and more feasible to outsource computations to the cloud while obtaining cryptographic guarantees about the correctness of their outputs. The most prominent protocols used today by far are Pinocchio [35] and the further optimized version proposed by Groth et al. [36], [37]. They are extremely efficient for practical applications and made their way already in many mainstream applications. The major drawback of this solutions for standard applications where the prover is in possession of the witness, is the "toxic waste" needed. This means that the system makes use of some global secure parameters which must be used in the setup phase but need to be deleted afterwards for secure operation. In fact, anybody in possession of the secure parameters to generate the so-called common reference string (CRS) can generate arbitrary proofs without knowing the witness.

To overcome the drawback for this type of SNARKs, novel systems have been proposed as presented in the next chapters. Nevertheless, novel extensions have also been developed which help to circumvent the problem for many application scenarios by making the CRS updateable and/or universal [38]–[40]. The main existing implementations of this approach are PySNARK[5] and libsnark[6].

Nevertheless, all work mentioned so far does not deal with the privacy of the inputs of the computation, therefore, it is natural to ask if this technology can be combined with MPC. Ideally in SlotMachine the privacy engine should be able to generate proofs for correct operation but does not have access to the inputs of the individual parties in plaintext. The only known proposal so far in this direction was by Schoenmakers in his Trinocchio system [41]. As its name suggests, it is based on Pinocchio [35], and allows for cryptographically verifiable computations, potentially with less effort than the computation itself. Moreover, in a follow-up proposal Pinocchio was made adaptive (or "hash-and-prove") and even combined in with ideas from Trinocchio [42] leading to an adaptive zero-knowledge protocol for computation independent commitments also compatible with MPC.

Although the current state of the art seems very promising it still has various problems to overcome for application in SlotMachine. It is unclear if and how to combine universal zk-SNARKS with MPC-based Pinocchio and if they could be lifted to be used with more efficient Groth version or another proof system in general. Allowing for a trustworthy —possibly also done by MPC— computation of a CRS also depends on the use case.

4.2.2 Discrete log-based systems

In [43] the authors provide a zero-knowledge argument called BCCGP-sqrt for arithmetic circuit satisfiability. It is based on an efficient zero-knowledge argument of knowledge of openings of two Pedersen multi-commitments satisfying an inner product relation. It has logarithmic communication complexity as well as logarithmic interaction and linear computation complexity for both the prover and the verifier.

[5] https://github.com/Charterhouse/pysnark
[6] https://github.com/scipr-lab/libsnark

A more versatile and extended version of the above has been presented in [44]. The protocol is called Bulletproofs and requires no trusted setup. Bulletproofs are used to show that an encrypted plaintext is well formed, e.g., that it is within a given margin or range without revealing information about the plaintext of the encrypted message. Contrary to SNARKs, Bulletproofs do not depend on a CRS generated from toxic waste, however, verification is costlier. Nevertheless, it supports proof aggregation with only *O(log(m))* overhead and better amortized cost.

Interestingly, Bulletproofs were developed to increase the privacy of cryptocurrencies, which is also relevant in the context of SlotMachines. By application of this technique, transaction can be proven valid without revealing money or token flows. Various implementations are mentioned on the homepage[7]. Additionally we found another useful version of Bulletproofs in the repository of Hyrax[8].

According to our analysis, Bulletproofs together with Pedersen commitments seem an alternative approach to the adaptive Pinocchio variant of zk-SNARKS. It does not require any trusted setup, range proofs are supported and can even be batched. However, it is unclear if it can be combined with MPC and more research is needed if selected for SlotMachine.

Hyrax [45] is another framework for proofs based on discrete logarithms and an extension to Bulletproofs. According to the authors, Hyrax's proofs are configured to be small and fast and a useful point in a large trade-off space for this type of proof systems. The software repository is located on GitHub[9].

4.2.3 Short PCP

Most prominent and available system from this category is called zk-STARK [46]. It is a zero-knowledge proof system that no longer relies on a trusted setup where the "toxic waste" parameters are initialized. This together with the long-term post-quantum security property make it a very attractive system for many applications. The name zk-STARK stems from the basic properties of the scheme. It is zero-knowledge, non-interactive, asymptotically optimal in efficiency and transparent.

Currently zk-SNARKs are roughly 1000x shorter than zk-STARK proofs, so they cannot easily replace SNARKs in all applications, it has to be ensured that the use case can support the overhead in size. Additionally, as discussed in [46], many of the alternative systems which have been implemented to realize zero knowledge proofs outperform zk-STARK for sufficiently small-size computations, for low-depth parallel computations, and/or for batched and amortized computations. Although this might not be a problem for the SlotMachine use case it seems that the hash-based nature of zk-STARK, which provides the post-quantum security, is problematic for MPC integration. We are not aware of any attempt to realize multi-stakeholder proofs with input privacy based on MPC or other methods. Therefore, this technology is not in the focus for PE integration but may be useful for related tasks in SlotMachine. A mature implementation of STARKs is available on GitHub[10].

[7] https://crypto.stanford. edu/bulletproofs/
[8] https://github.com/hyraxZK/bccgp
[9] https://github.com/hyraxZK/hyraxZK

[10] https://github.com/elibensasson/ libSTARK

4.2.4 Other Solutions

Another type of zero-knowledge proof system which is very efficient and relevant for verifiable computing is represented by ZKBoo [47] and ZKB++ [28]. They have been used efficiently in MPC-in-the-head computations, but we doubt that they can be easily integrated with basic MPC. They work on Boolean circuits, which is not what we need in SlotMachine. Implementations for ZKBoo[11] are available and ZKB++ is part of the picnic submission to the NIST post-quantum cryptography challenge[12]. They are also not succinct which makes them less efficient for the verification of larger computations and no adaptive variants have been proposed so far to work on committed data.

The same is true for Ligero [48], a lightweight sublinear argument of knowledge without a trusted setup. It is also a relatively simple protocol for NP with communication complexity proportional to the square-root in the circuit size. It has better performance for larger circuit sized than ZKB++ but is based on collision resistant hash functions, which renders it less useful in combination with MPC for input privacy and cannot be used in the PE.

4.3 Overview of Available Frameworks

Contrary to our analysis of MPC, which is based on intensive testing and benchmarking, for verifiable computing approaches we only did code review and analysis but no implementations. Benchmarking results will be added in later reports when dedicated tests for our specific use case and the respective MPC protocol have been done. Nevertheless, for our first review the following software packages have been analysed.

4.3.1 PySNARK

Summary: Library for programming zk-SNARKs directly in Python
Location: https://github.com/meilof/pysnark
Authors / Maintainer: Meilof Veeningen, Philips Research
Dependencies / Required Software • Python 3 • qaptools (optional backend) • libsnark (optional backend) • snarkjs (optional backend)
Advantages:

[11] https://github.com/Sobuno/ZKBoo
[12] https://github.com/Microsoft/Picnic

• Easy to use • Easy to set up (depending on backend) *Disadvantages:* • Little documentation
Development Status: Active, but only 1 developer (last commit June 2021)

4.3.2 libsnark

Summary: A C++ library for zkSNARK proofs
Location: https://github.com/scipr-lab/libsnark
Authors / Maintainer: SCIPR Lab
Dependencies / Required Software • GCC/Clang, CMake, GMP • libff • libfqfft • ate-pairing • xbyak • libsnark-SUPERCOP
Advantages: • De-facto standard in research community *Disadvantages:* • Many dependencies, difficult to set up • Not production-ready
Development Status: Somewhat active (last commit July 2020)

4.3.3 libSTARK

Summary: A library for zero knowledge (ZK) scalable transparent argument of knowledge (STARK)
Location: https://github.com/elibensasson/libSTARK
Authors / Maintainer: Eli Ben-Sasson, SCIPR Lab
Dependencies / Required Software • GCC, OpenMP
Advantages: • Under heavy development and progressing fast *Disadvantages:* • Not production-ready
Development Status: Inactive (last commit December 2018)

4.3.4 Bulletproofs

Summary: Bulletproofs are short non-interactive zero-knowledge proofs that require no trusted setup.
Location: https://crypto.stanford.edu/bulletproofs/ Various implementations exist: • https://github.com/apoelstra/secp256k1-mw/tree/bulletproofs: *Summary:* An implementation of Bulletproofs in C by Andrew Poelstra and Pieter Wuille. Uses constant time operation for proving and is very fast. Includes a tool for converting Pinocchio circuits to Bulletproof circuits and generating proofs for arbitrary statements. *Authors / Maintainer:* Andrew Poelstra *Development Status:* Unclear (project is branch of a fork of a fork) • https://github.com/bbuenz/BulletProofLib: *Summary:* An implementation of Bulletproofs in Java. Includes a general tool for constructing Bulletproofs for any NP language using the Pinocchio tool chain. Prototype code. *Authors / Maintainer:* Benedikt Bünz *Development Status:* Inactive, only one developer (last commit February 2019) • https://github.com/dalek-cryptography/bulletproofs: *Summary:* A pure-Rust implementation of Bulletproofs using Ristretto.

Authors / Maintainer: Henry de Valence, Cathie Yun, and Oleg Andreev (Dalek Cryptography)

Development Status: Somewhat active (last commit January 2020)

- https://github.com/hyraxZK/bccgp

  Summary: Independent re-implementations of two protocols due to Bootle et al. and Bünz et al.

  Authors / Maintainer: Riad S. Wahby

  Development Status: Inactive (Single commit in February 2018)

4.4 Evaluation Results

Finally, we give two quick overview tables found in [49] and [45]. They show the main properties of the different approaches which are important for selection in application design as discussed in the previous sections. They are shown in Figure 3 and Figure 4.

	prover scalability (quasilinear time)	verifier scalability (polylogarithmic time)	Transparency (public randomness)	Post-quantum security
hPKC	Yes	Only repeated computation	No	No
DLP	Yes	No	Yes	No
IP	Yes	No	Yes	No
MPC	Yes	No	Yes	Yes
IVC+hPKC	Yes	Yes	No	No
ZK-STARK	Yes	Yes	Yes	Yes

Figure 3: Theoretical comparison of universal realized ZK systems from [49].

	Short Proofs	Fast $\mathcal{P}$	Fast $\mathcal{V}$	Trusted setup?	Assumption
libsnark	✓	✗	✓	✗	Knowledge of exponent
Bulletproofs	✓	✗	✗	✓	discrete log
ZKB++	✗	✓	✗(ish)	✓	collision-resistant hashes
Ligero	✓(ish)	✓	✓(ish)	✓	collision-resistant hashes
libSTARK	✓	✗	✓	✓	Reed-Solomon conjecture

Figure 4: Comparison of ZK systems from Hyrax [23].

Performance measurements have been conducted for PySNARK/qaptools. To test the overall performance, we implemented basic tests. In Table 4 we summarize first results for the task of finding the maximum element in a list of varying size and producing a proof that the result is indeed the maximum element (time in seconds) - sizes of proof and verification keys remain constant at about 2,5 kB each. The first tests are very promising, and SNARKs could be a valuable tool in SlotMachine.

However, more research and implementation work is needed to learn more insights, especially for combination with MPC.

N	Data generation	Program execution	Setup (equations, keys)	Proof generation	Proof verification
1	0,032	0,001	0,186	0,035	0,038
10	0,054	0,008	1,351	0,165	0,039
100	0,285	0,063	11,940	1,280	0,047
1000	2,567	0,595	148,410	18,986	0,118

Table 4: Performance figures for proof systems.

5 Evaluation of Blockchains

5.1 Introduction

As we have already mentioned, our system should support the PBFT consensus, which enables managing the possible adversaries in the network. Moreover, we aim for a solution which is optimized for inter-organizational logic, flexible enough to accommodate changes, high-performance to compete with the major, non-BFT, consensus algorithms that exist today, such as etcd, zookeeper, consul, etc., all of that while providing greater resilience, security guarantees, and flexibility for application developers. Therefore, based on this, we have selected Tendermint, as a practical blockchain solution that supports PBFT consensus protocol. Tendermint has high performance and can achieve thousands of transactions per second (tx/s) on dozens of nodes distributed around the globe, with latency of about one second, and performance degrading moderately in the face of adversarial attacks [26]. In this section we present the results of the evaluations we have performed on the Tendermint scalability.

5.2 Tendermint evaluation

To the best of our knowledge, there is no publicly available evidence on the performance of Tendermint in case the number of nodes reaches hundreds or thousands. The performance testing with thousands of nodes would require a lot of computational resources and therefore would be expensive. We have performed some tests with 100 nodes. The tests are performed on the Amazon Cloud AWS EC2 machines, used for the Tendermint nodes. Tendermint is deployed using the testnets software (see https://github.com/informalsystems/testnets for more details), which internally uses Ansible and Terraform for the infrastructure orchestration and allocation of the required resources. The metrics calculated during the tests are stored in the Influx database and visualized using the Grafana UI.

The first experiment is performed with the test load of 10 tx/s sent to each of the 100 nodes within the 3 minutes time slot. The second and the third experiment are performed with the test load of

1000 tx/s. The number of the peer nodes, i.e., the nodes each node is connected to, for the first two experiments is 100 for node0 and 1 for the other nodes. That network topology is automatically chosen by Tendermint, based on the configuration for the number of inbound and outbound connections, which is in this case set to 100. In addition, no persistent peers, i.e., the peers to which the connection must be established independent of what is specified in the configuration file, are specified. In the third experiment, for each node we have specified 10 persistent peers, in the following manner: node1 -> node10, node20, node30, …, node100; node2 -> node11, node21, node31, …, node1; …; node100 -> node9, node19, node29, node39, …, node99. Additionally, for each node we set the maximum number of inbound and outbound peers to 10, so that each node can create 30 total connections at most (10 to the persistent peers + 10 for the inbound connections + 10 for the outbound connections). This network topology ensures better communication among the nodes, where each node can reach each other node in maximum 3 hops (in both cases when the network chooses the outbound connections for e.g. node1 to: either node5 or node6, node15, node25, …, node95, beside the connections to the persistent peers, each other node can be reached in maximum 3 hops). The figures below show the block interval time metric for each experiment. It represents the time needed to append a new block to the blockchain. As we can see from the figures, for the first experiment it is on average ~7s, for the second experiment ~30s, and for the third experiment ~3s. The specified network topology seems to largely influence the block interval time. In the first 2 experiments, node0 experienced the network congestion, since it was connected to all other nodes. The block interval time from the third experiment shows that nodes can still communicate with a relatively high performance.[13]

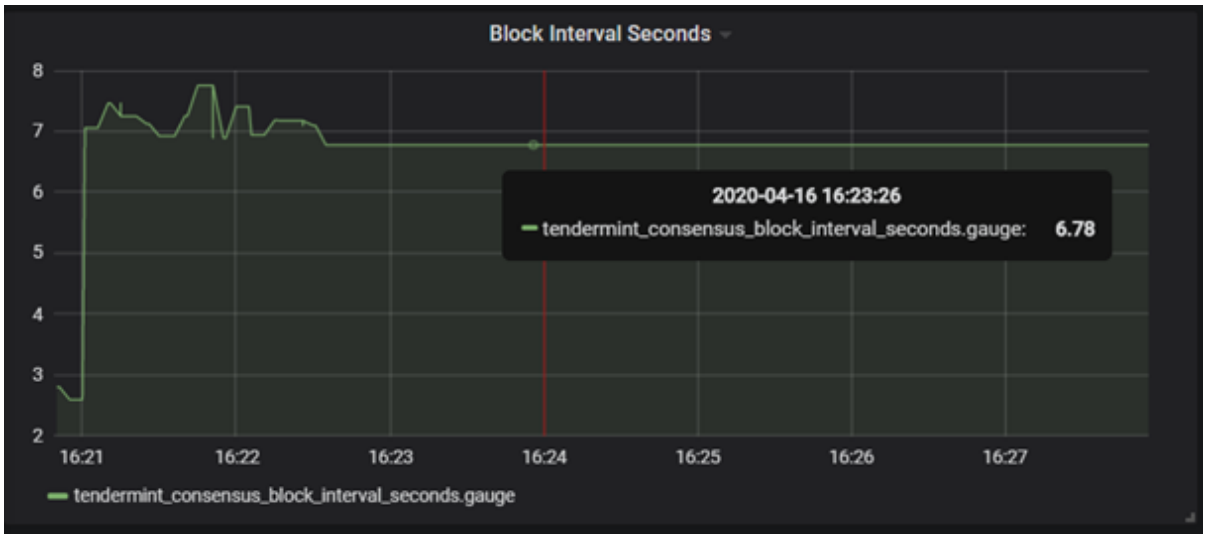

Figure 2: The block interval time for the first experiment.

[13] The block interval time might to a certain extent be influenced by the limited computational resources of the machines used for the nodes, which are, in this case, the Amazon t3.medium machines.

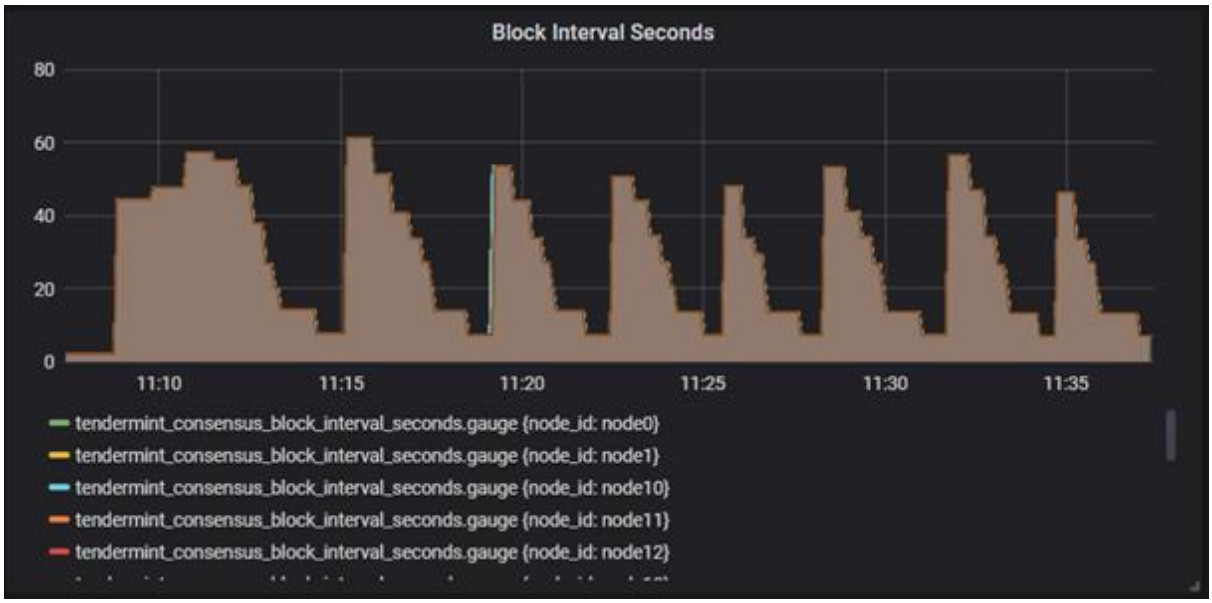

Figure 3: The block interval time for the second experiment

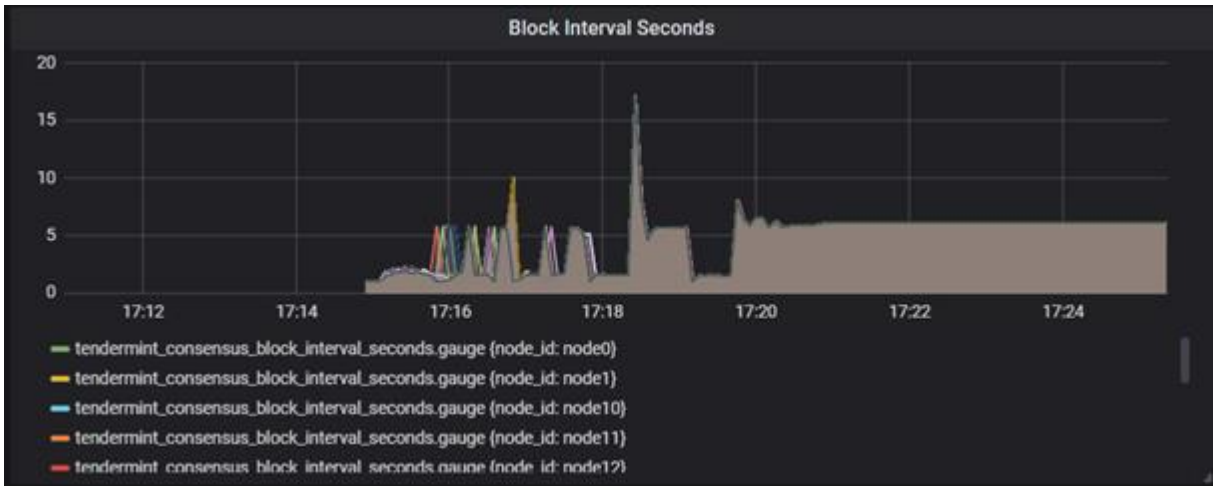

Figure 4: The block interval time for the third experiment

6 Conclusions and Recommendations

In our work on MPC we did intensive testing. The task turned out to be very time consuming with many frameworks to check for different properties. Moreover, the frameworks are often missing documentation and contain many suspected bugs and unexpected behaviour. For the time being, we found MP-SPDZ to be the most comprehensive solution and best suited for testing, and MPyC to be the most flexible one for rapid prototyping and testing. However, because results from one implementation cannot easily be mapped to another, we decided to implement core algorithms on both systems for comparison and consistency checking. In case of 2PC, we found ObliVM to be the most mature and reliable system, and we used it as a reference for benchmarking against garbled circuit approaches.

For the case of zk-SNARKS it turned out that this is a very dynamic field with lots of progress. We found many different implementations, with some being quite mature. However, when it comes to our specific use case with MPC integration, no proper framework was found, and no implementations are available.

Based on the findings in this report and benchmarking results we extracted the current set of recommendations to be followed in the upcoming project phase of SlotMachine.

- As already expected, the existing MPC systems lack scalability for direct application by AUs in the SlotMachine. It is not feasible to let every AU be an MPC node. Therefore, we recommend distributing the MPC system over 3-4 master nodes which can be statically or dynamically assigned by the PE.

- Given the first performance results we also expect that a passively secure implementation of this configuration achieves the performance goals expected by the requirements specification task of SlotMachine. This would be a good starting point for further exploration. Actively secure solutions are not required by the project, although new results show only minimal slowdown in the honest majority setting. Nevertheless, because we are aiming at a publicly verifiable MPC solution the active security would not add benefits in terms of robustness.

- To assure transparency we recommend starting from the approach on adaptive Pinocchio [42]. It seems well suited for our use case and integrates with MPC. Therefore, we will implement this protocol in a next step and identify the gaps for application in SlotMachine, once the platform functionality is fully specified. However, research should already be undertaken to circumvent the already known limitations based on the global CRS.

- Finally, the Tendermint blockchain solution selected turned out to scale very well. Results show that it is feasible to let every AU run a blockchain node if wished, i.e., a distributed ledger between all AUs is possible. As all parties are known and have to be registered the permissioned model is also a good fit.

Based on this finding, the currently envisioned architectural proposal for the integration of the presented technologies on a very high level is presented in Figure **5**. The flow resembles a blockchain based sealed-bid auction where bids (margins and weights in our case) are committed to on the blockchain in the first phase. However, instead of opening the inputs in the second phase, they are sent to the MPC system within the PE which computes (optimizes in our case) the best solution and

reveals the result together with a proof showing the improvement over previous flight sequences. Because the proof is zero-knowledge and all inputs are hidden we achieve privacy and correctness at the same time. Furthermore, all AUs will be able to verify the correctness on their own, increasing the trust in the SlotMachine platform. However, this is only the currently envisioned architecture and many research questions must be solved on the way, i.e., there are many technical risks associated with this approach. It is currently not clear if this full integration is possible efficiently and if all aspects are implementable with the given resources. The proposal also lacks the integration of the credit/token system currently under design to establish fairness and equity. Nevertheless, the presented architecture is very appealing from their properties and WP3 should work towards this direction.

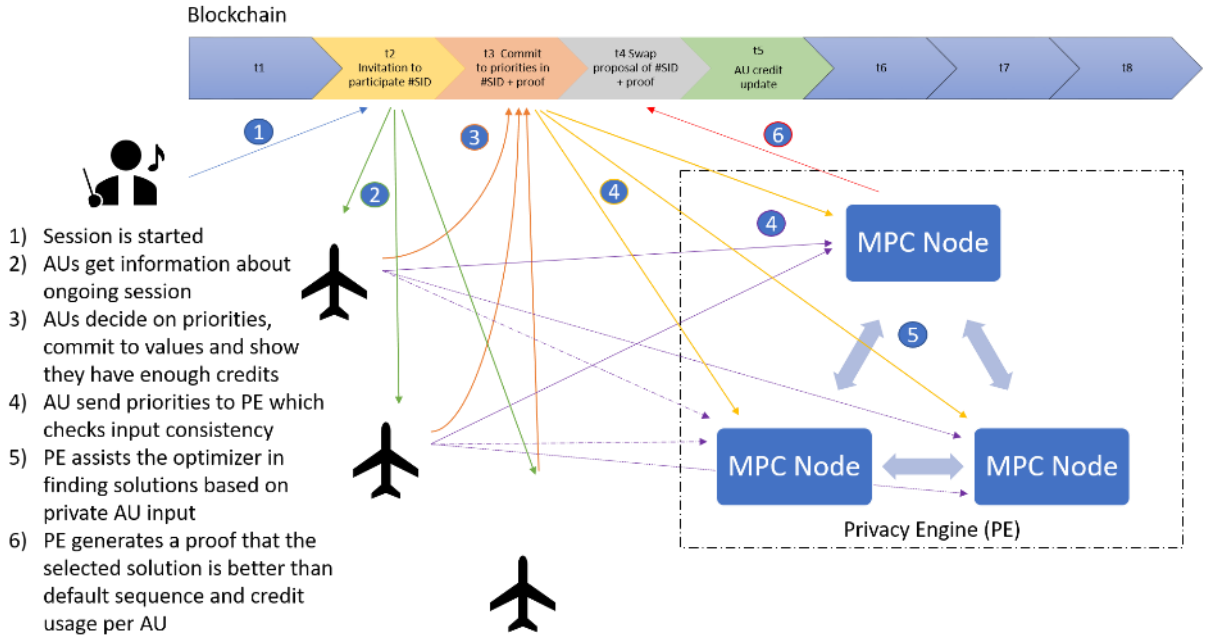

Figure 5: Proposal for the cryptographic

7 References

[1] N. Pilon, A. Cook, S. Ruiz, A. Bujor, und L. Castelli, „Improved Flexibility and Equity for Airspace Users During Demand-capacity Imbalance An introduction to the User Driven Prioritisation Process“, 2016. [Online]. Verfügbar unter: https://westminsterresearch.westminster.ac.uk/item/q2539/improved-flexibility-and-equity-for-airspace-users-during-demand-capacity-imbalance-an-introduction-to-the-user-driven-prioritisation-process

[2] P. Bogetoft *u. a.*, „Secure Multiparty Computation Goes Live“, 2009, S. 325–343. doi: 10.1007/978-3-642-03549-4_20.

[3] H. S. Galal und A. M. Youssef, *Verifiable Sealed-Bid Auction on the Ethereum Blockchain*. 2018.

[4] S. J. Rassenti, V. L. Smith, R. L. Bulfin, S. Rassenti, V. Smith, und R. L. Bulfin, „A Combinatorial Auction Mechanism for Airport Time Slot Allocation“, *Bell Journal of Economics*, Bd. 13, Nr. 2, S. 402–417, 1982.

[5] S. De Vries und R. V. Vohra, *Combinatorial auctions: A survey*, Bd. 15. 2003. doi: 10.1287/ijoc.15.3.284.16077.

[6] M. Ul Hassan, M. H. Rehmani, und J. Chen, „DEAL: Differentially Private Auction for Blockchain based Microgrids Energy Trading“, *IEEE Trans. Serv. Comput.*, S. 1–1, 2019, doi: 10.1109/TSC.2019.2947471.

[7] Y. Jiao, P. Wang, D. Niyato, und K. Suankaewmanee, „Auction Mechanisms in Cloud/Fog Computing Resource Allocation for Public Blockchain Networks“, *IEEE Trans. Parallel Distrib. Syst.*, Bd. 30, Nr. 9, S. 1975–1989, Sep. 2019, doi: 10.1109/TPDS.2019.2900238.

[8] Y. Jiao, P. Wang, D. Niyato, und Z. Xiong, „Social Welfare Maximization Auction in Edge Computing Resource Allocation for Mobile Blockchain“, in *2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO, Mai 2018, S. 1–6. doi: 10.1109/ICC.2018.8422632.

[9] M. F. Franco, E. J. Scheid, L. Z. Granville, und B. Stiller, „BRAIN: Blockchain-based Reverse Auction for Infrastructure Supply in Virtual Network Functions-as-a -Service“, in *2019 IFIP Networking Conference (IFIP Networking)*, Warsaw, Poland, Mai 2019, S. 1–9. doi: 10.23919/IFIPNetworking.2019.8816843.

[10] H. S. Galal und A. M. Youssef, „Verifiable Sealed-Bid Auction on the Ethereum Blockchain“, in *Financial Cryptography and Data Security*, Berlin, Heidelberg, 2019, S. 265–278. doi: 10.1007/978-3-662-58820-8_18.

[11] E.-O. Blass und F. Kerschbaum, „Strain: A Secure Auction for Blockchains“, Springer, Cham, 2018, S. 87–110. doi: 10.1007/978-3-319-99073-6_5.

[12] I. A. Omar, H. R. Hasan, R. Jayaraman, K. Salah, und M. Omar, „Implementing decentralized auctions using blockchain smart contracts“, *Technological Forecasting and Social Change*, Bd. 168, S. 120786, 2021, doi: https://doi.org/10.1016/j.techfore.2021.120786.

[13] D. C. Sánchez, *Raziel: Private and Verifiable Smart Contracts on Blockchains*. 2017.

[14] A. Kosba, A. Miller, E. Shi, Z. Wen, und C. Papamanthou, „Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts“, in *2016 IEEE Symposium on Security and Privacy (SP)*, 2016, S. 839–858. doi: 10.1109/SP.2016.55.

[15] H. S. Galal und A. M. Youssef, „Succinctly Verifiable Sealed-Bid Auction Smart Contract“, in *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2018 International Workshops, DPM 2018 and CBT 2018, Barcelona, Spain, September 6-7, 2018, Proceedings*, 2018, Bd. 11025, S. 3–19. doi: 10.1007/978-3-030-00305-0_1.

[16] E. Ben-Sasson, A. Chiesa, E. Tromer, und M. Virza, „Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture“, in *23rd USENIX Security Symposium (USENIX Security 14)*, San Diego, CA, Aug. 2014, S. 781–796. [Online]. Verfügbar unter: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/ben-sasson

[17] M. Andrychowicz, S. Dziembowski, D. Malinowski, und L. Mazurek, „Secure Multiparty Computations on Bitcoin“, in *2014 IEEE Symposium on Security and Privacy*, Mai 2014, S. 443–458. doi: 10.1109/SP.2014.35.

[18] I. Bentov und R. Kumaresan, „How to Use Bitcoin to Design Fair Protocols“, 2014, S. 421–439. doi: 10.1007/978-3-662-44381-1_24.

[19] R. Kumaresan, V. Vaikuntanathan, und P. N. Vasudevan, „Improvements to Secure Computation with Penalties“, in *Proceedings of the 2016 {ACM} {SIGSAC} Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, 2016, S. 406–417. doi: 10.1145/2976749.2978421.

[20] R. Kumaresan und I. Bentov, „Amortizing Secure Computation with Penalties“, in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2016, S. 418–429. doi: 10.1145/2976749.2978424.

[21] F. Vogelsteller und V. Buterin, „EIP-20: ERC-20 Token Standard“, Nov. 2015, [Online]. Verfügbar unter: https://eips.ethereum.org/EIPS/eip-20

[22] L. Oliveira, L. Zavolokina, I. Bauer, und G. Schwabe, „To Token or not to Token: Tools for Understanding Blockchain Tokens“, 2018, doi: 10.5167/UZH-157908.

[23] S. Martins und Y. Yang, „Introduction to Bitcoins: A Pseudo-Anonymous Electronic Currency System“, in *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*, USA, 2011, S. 349–350.

[24] G. Maxwell, „CoinJoin: Bitcoin privacy for the real world“, 2013.

[25] N. Van Saberhagen, *CryptoNote v 2.0*. 2013.

[26] E. Buchman, „Tendermint: Byzantine fault tolerance in the age of blockchains“, PhD Thesis, 2016.

[27] E. Androulaki, J. Camenisch, A. D. Caro, M. Dubovitskaya, K. Elkhiyaoui, und B. Tackmann, „Privacy-preserving auditable token payments in a permissioned blockchain system“, in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, New York NY USA, Okt. 2020, S. 255–267. doi: 10.1145/3419614.3423259.

[28] R. Cramer, I. B. Damgard, und J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*. Cambridge: Cambridge University Press, 2015. doi: 10.1017/CBO9781107337756.

[29] D. Evans, V. Kolesnikov, und M. Rosulek, „A Pragmatic Introduction to Secure Multi-Party Computation“, *Foundations and Trends® in Privacy and Security*, 2018, doi: 10.1561/3300000019.

[30] Y. Lindell und A. Nof, „A Framework for Constructing Fast MPC over Arithmetic Circuits with Malicious Adversaries and an Honest-Majority“, in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security - CCS '17*, New York, New York, USA, 2017, S. 259–276. doi: 10.1145/3133956.3133999.

[31] K. Chida *u. a.*, „Fast Large-Scale Honest-Majority MPC for Malicious Adversaries“, Springer, Cham, 2018, S. 34–64. doi: 10.1007/978-3-319-96878-0_2.

[32] P. S. Nordholt und M. Veeningen, „Minimising Communication in Honest-Majority MPC by Batchwise Multiplication Verification“, Springer, Cham, 2018, S. 321–339. doi: 10.1007/978-3-319-93387-0_17.

[33] J. Furukawa und Y. Lindell, *Two-Thirds Honest-Majority MPC for Malicious Adversaries at Almost the Cost of Semi-Honest*. 2019.

[34] C. Baum, I. Damg\aard, und C. Orlandi, „Publicly Auditable Secure Multi-Party Computation“, Springer, Cham, 2014, S. 175–196. doi: 10.1007/978-3-319-10879-7_11.
[35] B. Parno, J. Howell, C. Gentry, und M. Raykova, „Pinocchio: Nearly Practical Verifiable Computation“, in *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, 2013, S. 238–252. doi: 10.1109/SP.2013.47.
[36] J. Groth, *On the Size of Pairing-based Non-interactive Arguments*. 2016.
[37] J. Groth und M. Maller, „Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs“, in *Advances in Cryptology – CRYPTO 2017*, Cham, 2017, S. 581–612.
[38] M. Maller, S. Bowe, M. Kohlweiss, und S. Meiklejohn, *Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updateable Structured Reference Strings*. 2019.
[39] A. Kosba, D. Papadopoulos, C. Papamanthou, und D. Song, *MIRAGE: Succinct Arguments for Randomized Algorithms with Applications to Universal zk-SNARKs*. 2020.
[40] J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, und I. Miers, *Updatable and Universal Common Reference Strings with Applications to zk-SNARKs*. 2018.
[41] B. Schoenmakers, M. Veeningen, und N. de Vreede, „Trinocchio: Privacy-Preserving Outsourcing by Distributed Verifiable Computation“, Springer, Cham, 2016, S. 346–366. doi: 10.1007/978-3-319-39555-5_19.
[42] M. Veeningen, „Pinocchio-Based Adaptive zk-SNARKs and Secure/Correct Adaptive Function Evaluation“, Springer, Cham, 2017, S. 21–39. doi: 10.1007/978-3-319-57339-7_2.
[43] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, und C. Petit, „Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting“, 2016. doi: 10.1007/978-3-662-49896-5_12.
[44] B. Bunz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, und G. Maxwell, „Bulletproofs: Short Proofs for Confidential Transactions and More“, 2018. doi: 10.1109/SP.2018.00020.
[45] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, und M. Walfish, „Doubly-Efficient zkSNARKs Without Trusted Setup“, 2018. doi: 10.1109/SP.2018.00060.
[46] E. Ben-Sasson, I. Bentov, Y. Horesh, und M. Riabzev, *Scalable, transparent, and post-quantum secure computational integrity*. 2018. [Online]. Verfügbar unter: https://eprint.iacr.org/2018/046
[47] I. Giacomelli und C. Orlandi, „ZKBoo: Faster Zero-Knowledge for Boolean Circuits“, *Usenix Security*, 2016.
[48] S. Ames, C. Hazay, Y. Ishai, und M. Venkitasubramaniam, „Ligero: Lightweight sublinear arguments without a trusted setup“, 2017. doi: 10.1145/3133956.3134104.
[49] S. Setty, *Spartan: Efficient and general-purpose zkSNARKs without trusted setup*. 2019.


SlotMachine


SESAR
JOINT UNDERTAKING


AIT
AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY


FREQUENTIS
FOR A SAFER WORLD


JKU
JOHANNES KEPLER
UNIVERSITÄT LINZ


SWISS