Auditable Serverless Computing for Farm Management

Servio Palacios
spalacio@purdue.edu
Purdue University, Qlever LLC
West Lafayette, Indiana, USA

Drew Zabrocki
dz@centricity.us
Centricity Global
Seatle, Washington, USA

Bharat Bhargava
bbshail@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Vaneet Aggarwal
vaneet@purdue.edu
Purdue University
West Lafayette, Indiana, USA

ABSTRACT

Currently, a multitude of applications that target farm management activities has been proposed. Unfortunately, those applications do not interoperate and require that the farmer utilize several of them, making operations and task management cumbersome. Similarly, those applications require a tiered pricing model ranging from a restricted/limited free tier to several thousand dollars per year—mainly in a software as a service SaaS model.

In this paper, we propose a novel mix of serverless functions, shared ledgers, webhooks, and REST APIs to enhance Agriculture/-Farm Management Systems, providing an integrated solution for Task, User, and Field Management that exploits a fine-grained pricing model. Further, our technique utilizes *serverless oblivious smart contracts* as a building block. To the best of our knowledge, this is the first solution that leverages serverless functions and shared ledgers to provide an elastic, pay-as-you-go, and auditable task management system for the AG industry.

Our work has a significant impact on providing an open-source solution released and used in production that can pave the way for future relevant ideas in the AG industry. Therefore, we demonstrate the system feasibility exposing evidence of the system performance for auditable task creation and chat mirroring.

CCS CONCEPTS

• **Applied computing** → **Agriculture**.

KEYWORDS

Auditable Serverless Computing, Serverless Computing, Farm Management, Serverless Functions, Serverless Oblivious Smart Contracts, Auditable Farm Management

ACM Reference Format:
Servio Palacios, Drew Zabrocki, Bharat Bhargava, and Vaneet Aggarwal. 2021. Auditable Serverless Computing for Farm Management. In *Big Data in Emergent Distributed Environments (BiDEDE'21), June 20, 2021, Virtual Event, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3460866.3461770

BiDEDE'21, June 20, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8465-0/21/06…$15.00
https://doi.org/10.1145/3460866.3461770

1 INTRODUCTION

There are an extensive set of applications that target automating farm business processes. Solutions such as Granular [9] utilize data to determine the profitability of fields under certain crops; their monetization model is based per acre. Conservis [6] aims to manage field activities such as applications and planting and analyze yields. Farmbrite [7] targets record-keeping, crop planning, scheduling, and activity tracking. Agriculture ERP systems such as FarmLogics [8] and Microsoft Dynamics [10] provide modules for crop management, Geographic Information Systems, and Task/Work Flow Management. Those systems are far from a customized solution that fits every customer's requirements or budget. In particular, current solutions require investing in mostly yearly subscriptions without considering the time that modules or processing is not needed or is idle. Further, these systems assume that all modules utilize the same processing units or pricing model. Also, many of those solutions do not provide auditability features for task management. Most of the applications reviewed are delivered as a 'software service' (SaaS) and offer tiered pricing running from a limited free tier to several thousand dollars per user per year. Similarly, most of those applications collect user data in the cloud, leading to significant privacy issues [13, 14].

In this paper, we propose a novel mix of serverless functions, shared ledgers, webhooks, and REST APIs to enhance Agriculture/-Farm Management Systems, providing an integrated solution for Task, User, and Field Management that exploits a fine-grained pricing model named AuditFarm.io. Our technique leverages a relaxed version of *serverless oblivious smart contracts* [13, 14] as a building block. For instance, AuditFarm.io leverages Twilio Serverless Functions [16] to provide a straightforward communication channel with people across the farm operation, specifically: crop advisors, applicators, irrigators, and administrative staff. The objective is to streamline real-time communication in a way that is useful and easy to adapt while systematically logging events for immediate and historical analysis. We accomplished a customized and auditable Task management system via Azure Serverless Functions, Azure Key-Vault, and IBM Hyperledger Fabric. Our solution exploits the main characteristics of serverless computing, i.e., elasticity and pay-as-you-go. The user is charged only for the utilized resources.

1.1 Problem Definition

Farmers often use a dozen or more software solutions; each system offers benefits; however, the value that could be achieved from data-enriched decisions is often trapped in siloed systems that do not interoperate.

Field and administrative staff constantly switch between different applications making it difficult to maintain focus and remember where to store certain information. Redundant data entry and transcription introduce additional labor cost and is error-prone; the greatest disadvantage is lost time and lack of clear communication among the team. When faced with pressing challenges of logistics and agronomy, interaction with data and systems often are put aside to address issues staring them in the face.

Software and solution providers have attempted to resolve these challenges by either developing more features—swiss army knife approach— or through proprietary integrations, which often fall short on delivering optimal user experiences or require significant ongoing investments to implement and maintain.

Partially connected systems often result in user dissatisfaction with the spider web of applications and endless forms. Grasping for 'what works,' users often gravitate towards simplified communications and familiar tools that work for all, such as Text Messaging and To-Do Lists. While easy to use and broadly accessible, these simplified communication tools lack data standardization making reporting and analysis nearly impossible; unfortunately, the information captured is often lost.

1.2 Overview of technical approach

In this project, we enhanced the Task Management System through a set of serverless functions that allow task creation, user assignment, field geolocation, and chat mirroring (Figure 1). The Parsing Module (§3.1.1) integrates SendGrid's Inbound Parser to provide email parsing guarantees that can send an Excel File through a webhook that triggers an Azure Function's execution. These Azure Functions create Field Tasks and assign specific users. The Azure Functions leverages Google Maps API and ClickUp Field Location custom fields to expose real-time field monitoring capabilities. The Task Management module (§3.1.3) allows the use of ClickUp automations, webhooks, chat views, users, and custom fields to provide Task Management and real-time monitoring features. The Chatbot engine (§3.1.2) integrates Twilio serverless functions, current SMS Text Messaging groups, and ClickUp chat views to offer an integrated tool that allows monitoring issues in the field in real-time.

1.3 Contributions

The contributions of this paper are summarized as follows:

- We develop an integrated solution that provides auditable task management capabilities for farm management.
- As a fundamental component of our implementation, we include the use of serverless functions to provide a diverse use of compute nodes (elasticity) and a new pricing model for the AG industry. Additionally, we incorporate privacy-preserving design principles as described in [13].
- We contribute an open-source implementation of our proposed technique that can serve future solutions in the AG industry[1].

2 TECHNICAL BACKGROUND

This project provides Task Management Automation through a set of short-lived serverless functions that automate task creation, user assignment, and field management. Similarly, this work exposes a chatbot functionality via Twilio Serverless Functions to mirror Group Text Messaging to the Farm Management Web Application. We first explain Serverless Computing, and then we explore the core concepts for accountability, traceability, and ordering of tasks.

2.1 Serverless Computing

In this section, we utilize Castro, Ishakian, Muthusamy, Slominski (2019) contributions [1]. Serverless Computing is defined as "a platform that hides server usage from developers and runs code on-demand automatically scaled and billed only for the time the code is running" [1]. Castro et al. described elasticity and cost —pay-as-you-go— as the two critical components of serverless computing. Serverless Computing diminishes IT administration's burden on virtual machines and resources, facilitating business application development. For instance, a developer can expend a significant amount of time allocating the correct virtual machines, resources, and subscriptions to provide a particular business application. These business applications can be reduced to stateless, self-contained computation kernels that run eventually in a bursty fashion for some specific use cases. The bursty nature of serverless functions requires that resources are allocated according to demand—elasticity; this property must scale down to zero and scaling up towards infinity. Similarly, a new cost model arises with serverless computing allowing the consumer to pay only for the resources they consume when a function is running—the pay-as-you-go cost model. Current activities for farm management often exhibit a bursty nature, i.e., seasonal tasks/applications making the use of serverless computing a natural computing model. We leverage serverless computing to enhance the Oblivious Smart Contract concept (§2.3) to provide an elastic compute engine and pay-as-you-go pricing model.

2.2 Blockchain technologies

Blockchain technologies have been widely used to provide features such as traceability, auditability, accountability, and event ordering, e.g., in the supply chain. In this section, we explain the main characteristics of shared ledgers that allow those features. Satoshi Nakamoto (2008) introduced the first concept of blockchain, proposing a solution for the double-spending problem named Bitcoin [12]. Blockchains are also called shared-ledgers or decentralized and immutable shared-ledgers, as explained in [1, 12]. The Bitcoin paper describes how a resource is not permitted to be consumed/spent twice by transactions, even in the presence of malicious actors. Therefore, Bitcoin introduces proof-of-work—based on cryptographic hash functions—to generate an immutable record. Utilizing cryptographic hashes and imposing complex constraints, the network

[1] As part of our contributions, we reported several bugs in well-known Task Managements systems. Moreover, some of these bugs have been fixed during the review process for this paper.

participants must compete to commit a transaction in the shared ledger. Moreover, the proof-of-work and the longest correct chain expose verifiable proof of the ledger's events—assuming that attackers or malicious nodes cannot control more than 50% of the nodes/participants, which will make it extremely hard to produce a parallel longest malicious chain. The proof-of-work and the inherently byzantine fault tolerance provided by a blockchain [2] provides a *verifiable* sequence of events and, therefore, is utilized to provide auditability, traceability, and accountability features.

AuditFarm.io utilizes a blockchain-gateway to provide asynchronous and anonymous accesses to shared ledgers. We store a hash of the content of a task or a set of tasks in the share-ledger so that we can verify the original content when running payment processes. AuditFarm.io implementation includes an IBM Hyperledger Fabric, but the blockchain-gateway can be extended to connect to other popular shared ledgers such as Ethereum and many others.

2.3 Oblivious Smart Contracts

An Oblivious Smart Contract (OSC) is a certified/pre-approved software code that computes on top of private data to obtain a privacy-preserving derivative from the private data, such as a pass/fail outcome [13, 14]. In the OSC trust model, the data owner preserves data ownership and approves the algorithm that will run on top of their private data (OSC.) This paper extends the OSC capabilities utilizing serverless functions, i.e., *serverless oblivious smart contracts* (SOSCs). The farmer approves in advance the algorithm that creates the tasks in the task management system. Further, the SOSC will keep track of the content of the task storing a cryptographic hash of the content of the task (or a set of tasks) in the ledger.

3 METHOD: SYSTEM MODEL

3.1 Architecture

Our architecture considers three critical modules: the Parsing Module, the Farm Management Module, and the Chatbot engine.

3.1.1 Parsing Module. The parsing module includes the integration of SendGrid's Inbound Parse [15] and a Serverless Function that takes the content sent by Sendgrid. Figure 1 shows the workflow with the following steps:

- An Administrator or Business user prepares a set of tasks in an Excel File Format.
- The administrator or business user emails the Excel File.
- SendGrid's Inbound Parse processes the email and Excel file.
- SendGrid triggers a webhook that points to an Azure Serverless Function that contains the semantics to create new tasks, assign users to tasks, and georeference the activities per field.
- The serverless function uses REST requests to the Task Management Module to create Tasks, assign users, and georeference fields.

We preserve the previous workflow utilized widely by the farmer, but we enhanced the operations by providing an automated Task Management module. The SendGrid Inbound Parsing requires modifications in the DNS records [15] to allow this feature. The administrator periodically runs this module—usually once every week, following the bursty nature of serverless computing.

Blockchain Gateway: The blockchain gateway stores the integrity (hash of the task's content) of parsed tasks for particular dates. Since the user can change the description of a task in the Task Management App, AuditFarm.io uses this gateway to check for the task's integrity and expose a trustworthy payment process.

3.1.2 Chatbot engine. In Figure 1, we observe that current Text Group Messaging (SMS groups) are integrated into the Farm Management Module via Twilio serverless functions [16] that mirrors all communications happening on the field, e.g., crop advisors, applicators, irrigators, and administrative staff. The business user can verify all history of issues and comments related to a particular field or activity.

3.1.3 Farm Management Module. The core component of this module is ClickUp [4]. ClickUp defines a hierarchy that includes Teams, Workspaces, Lists, and Tasks (Figure 1). The Parsing Module creates all Field Irrigation and Application Tasks utilizing ClickUp's REST API [5]. This module allows Administrator users to assign Tasks and comments to Irrigators or Applicators on the field. Irrigators and Applicators can update the assigned Tasks, e.g., setting a task as completed or attaching a photo of an issue on the field. Another essential component of the Farm Management Module is the Chat View capabilities. AuditFarm.io integrates currently available Text Group messaging into the ClickUp Chat Views. We created a Twilio Serverless function to mirror all the communications that happen in the field and with the administrator. The administrator or business user can verify all history of comments at a later time.

4 USE CASES

4.1 Clearly communicating work and job status from the office to the field and back

Office staff are entering irrigation and nutrient applications into spreadsheets and emailing them to staff and contractors on a weekly basis. Workers are then writing as-applied data on notebooks and returning paper records for data entry on an intermittent basis. These records are required for compliance and marketing purposes but could be most useful to agronomists if available on a near-real time basis. Streamlining the process without introducing additional steps or complex workflows is essential to engaging users who are mobile and often hurried to accomplish their tasks. Increasing requirements for sustainability and food safety data reporting require more complete documentation and auditable accounts of field activity.

We will 'meet users where they are'. Excel spreadsheets continue to be used to document and report activities. Text messages will be used to communicate status updates. When tasks are assigned for the week, office staff will email the serverless function which will automatically parse and assign tasks individually to each worker. Serverless functions will notify users via text message and update the digital 'to-do' list. Users will receive reminders to update their to-do lists. When complete, the to-do list will be processed by a serverless function to update the excel spreadsheet with the as-applied data which can then be reported. Meanwhile, agronomists and managers will have real-time work and agronomic data available without having to constantly follow-up with staff.

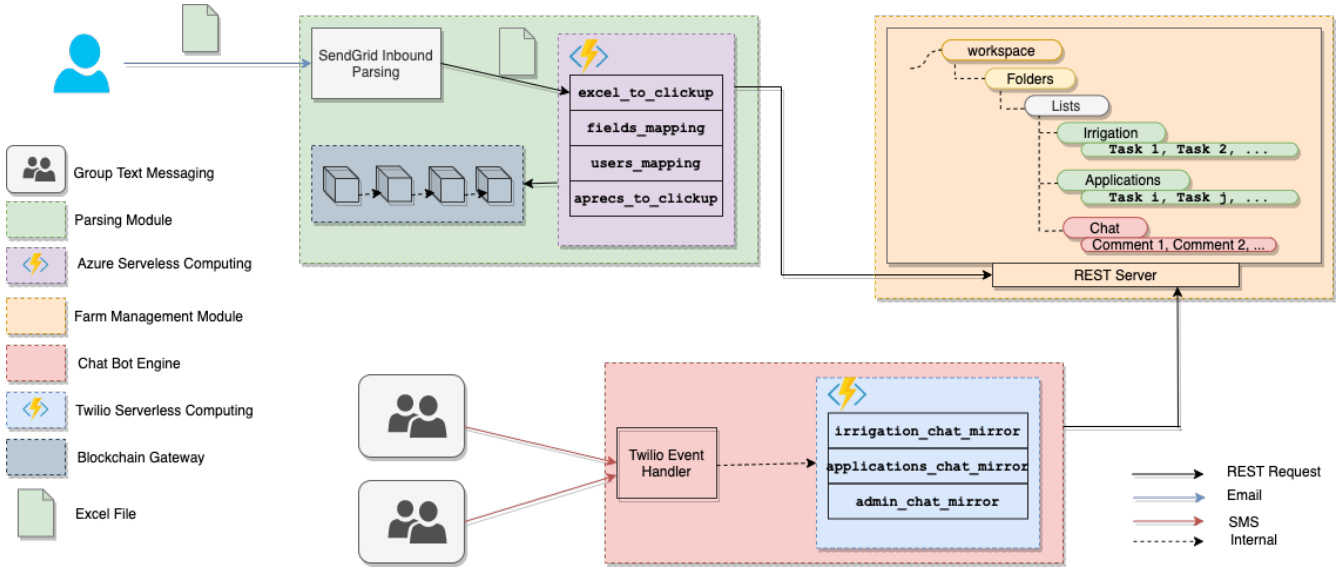

Figure 1: Auditable Farm Management with Serverless Functions. Architecture Overview.

4.2 Orchestrating internal and external business partners, using various systems, to execute nutrient, irrigation, and pest management jobs

Skone & Connors Ranching (SCR) operates across a vast territory and works with a number of suppliers and contractors to advise, manage and execute on applications of pest management and nutrient products in a sustainable and efficient manner. This requires coordination of dozens of third party companies, independent contractors, employees, and administrative staff.

SCR subscribes to ApRecs a unified recommendation and application management system. While all third-party data formats integrate seamlessly, encouraging third parties to update status in a separate system has proven difficult. While some contractors have adopted the ApRecs solutions, some have are tied with legacy systems. For those, ApRecs employs an API to enable interoperability with their mobile and web applications.

By leveraging serverless functions, updates to ApRecs will automatically create SMS messages and update to-do lists so that all parties are on the same page. ApRecs users will benefit from improved, real-time interaction with 3rd party companies. The business will also see reduced licensing costs since third-party users will not need individual accounts.

5 PROTOTYPE IMPLEMENTATION

Our project includes a released version ($v1.0$) moving into production by the end of this month[2]. All projects are hosted under GitLab. In future releases, we will publish a library for the Task Management System and Chatbot[3].

[2]March, 2021.
[3]All projects are hosted under GitLab and are kept private during this paper review

5.1 Parsing module

The parsing module includes the implementation of Azure Functions for parsing and Excel file sent by the SendGrid Inbound Parse Module. SendGrid points to our Azure Serverless Function (webhook call) and triggers the serverless function when an email with particular characteristics is received. SendGrid takes care of Spam Messages. We utilize an *HMAC* of the content of the Excel File as part of the flow. The key (used by $HMAC_{key}(content)$) is only known to the business user—admin that sends the email and the serverless function—via Azure Key Vault. To build, test and deploy the serverless function, we utilized Visual Studio Code $v1.54$ with Azure Functions 1.3.0 extension.

5.1.1 Fields dataset. We upload a set of currently used fields parsing an Excel File provided by the farmer. Since the amount of fields is reduced —less than 100 fields per season, it is a manageable number to store in the task management system. The field location assignment into a ClickUp field location *custom type* is done via Google Maps API. We exposed a Fields' Search Engine in the serverless functions to geo-reference the tasks. This process includes application records attributes to interact with other systems.

5.1.2 Users dataset. The users' dataset is obtained from an Excel file provided by the farmer. We follow a similar process as described above (users instead of fields.) This dataset is utilized by Twilio Serverless functions, i.e., to map phone numbers to real names in the Task Management System. Similarly, the Azure Serverless functions for parsing tasks assign the users in charge of a particular task using this users' dataset.

5.1.3 Blockchain Gateway. We implemented a Blockchain-Gateway utilizing IBM Hyperledger Fabric using JavaScript and Visual Studio Code. Table 1 shows the schema stored in the blockchain. We store a hash of a task's content (or a set of tasks) in the share-ledger to verify the original content when running payment processes.

Property(fabTask)	Type
id	String (UUID)
taskHash	String

Table 1: Task business network in IBM Hyperledger Fabric.

5.2 Task Management module

This module required the ClickUp REST API *v*2.0. We used TypeScript *v*4.0.3 and Axios *v*0.21.0. Additionally, we leverage ClickUp webhooks and automations to enrich ClickUp functionality.

ClickUp REST API limitations: We encountered several bugs and limitations with the ClickUp REST API during the development of the Task Management Module[4]. Please refer to Table 2.

5.3 Chatbot engine

To include the current SMS Text Message infrastructure, we develop a set of Twilio serverless functions utilizing Javascript (Table 3). The Twilio serverless functions mirror all messages into the Task Management Module. To build, test and deploy Twilio serverless functions, we utilized Visual Studio Code *v*1.54 with VSCode Twilio 0.1.0 extension.

6 EVALUATION

6.1 Experiment setup

We prepared a set of experiments to demonstrate serverless compute engines' performance that provides an integrated Task, Users, Fields Management System. We configured two Functions App and Key Vault resources under an Azure development subscription[5]. We setup a MacBook Pro (15-inch, 2017) with an Intel Core i7 2.8GHz and 16GB of RAM running macOS Catalina Version 10.15.4 and IBM Blockchain Platform 1.0.31 Visual Studio Code Extension as the Blockchain-Gateway.

6.2 Task Management Serverless Function Performance

We analyze the time to execute a serverless function that creates tasks into the Task Management System utilizing a REST API. We run 100 REST requests to the Azure serverless function. As explained in Section §5.1, the function will parse an Excel File and create the tasks accordingly. The process includes the creation of a task and a subtask (fertigation.) The serverless function includes users' and fields' search engine and field location assignment. In Figure 2, we can observe that the total time to execute a call is around 2516*ms*. The first calls to the Azure function (utilizing SendGrid's webhooks) take significantly more time to execute (3777*ms*).

The Parsing Module asynchronously accesses a shared ledger (IBM Hyperledger Fabric) to provide auditability features for the Task Management Module. In Figure 3, we can observe the different operations executed under the Blockchain. For instance, creating a new record in the ledger takes around 2.2*s*. Creating new records in the ledger is an asynchronous process.

[4]As part of our contributions, we reported several bugs in well-known Task Managements systems. Moreover, some of these bugs (Task Description Table) have been fixed during the review process for this paper.

[5]All Virtual Machines, subscriptions, and resources are provided by Centricity Global.

Figure 2: Parsing Module Azure Serverless Function Performance. The total time to execute the serverless function stabilizes after the second call.

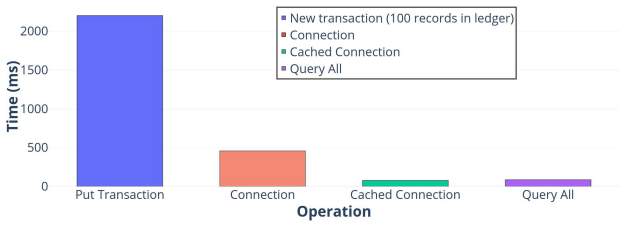

Figure 3: Parsing Module Blockchain-Gateway Performance. We asynchronously access a shared ledger to provide auditability features for the Task Management. Blockchain-Gateway is interacting with IBM Hyperledger Fabric and *taskContract* developed in JavaScript.

In Figure 4, we analyze the primary task management operations. This experiment include a set of test cases using the Chai assertion library [3], the Mocha test framework [11], and Axios. First, we analyze the task creation time (575*ms* on average); this operation includes searching and assigning users to the task and geolocating the field name into a Field Location. Similarly, we analyze the time to retrieve users (351*ms*) and fields (1202*ms*) from the task management system. We also show the time to create subtasks (around 150*ms*) and to log the activity (166*ms*) into the workspace.

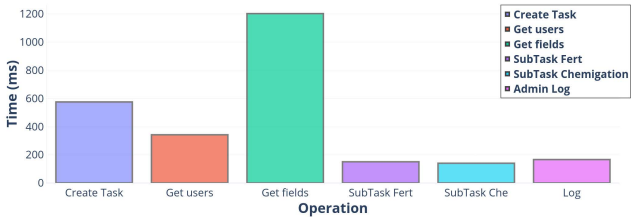

Figure 4: Task management REST API performance.

6.3 Application Records - Task Creation Performance

Similarly, we analyze the total time to create Application Record Tasks in the Task Management system. The running time for this

Bug | Limitation	Explanation
Task Description Table	A table created using Markdown through the ClickUp REST API is not shown in the mobile ClickUp
Field Tags	Tags created using the ClickUp REST API are not shown in the mobile ClickUp app
Field Location - Custom Fields	The ClickUp REST API does not allow updating Field Location custom Fields

Table 2: Summary of bugs and limitations in the ClickUp REST API. We notified ClickUp Team about this bugs since January, 2021. We tested with the Mobile App $v3.4.1$.

Serverless Function	Chat Mirror
twilio-chat-view-mirror/random-id-1	Irrigation
twilio-chat-view-mirror/random-id-2	Applications
twilio-chat-view-mirror/random-id-3	Admin

Table 3: Summary of Twilio Serverless functions for allowing Chat Mirrors in the Task Management System.

operation is 2.55s approximately. This time measures the total time to call an Azure Function via a webhook and the server's response time. The serverless function includes users' and fields' search engine and field location assignment.

6.4 Chatbot Serverless Function Performance

In this experiment, we analyze the total time that a text message takes to arrive at a Task Management System Chat View Mirror. We used cURL to test the response time from Twilio Serverless Functions. The average time for a call to a chat mirror function was 1.741678s. We tested all Twilio Serverless Functions in Table 3.

7 RELATED WORK

7.1 Farm Management Systems

We include the most relevant existing technologies for Farm Management Systems. Due to space limitations, we created a medium article[6] as an addendum to this section.

7.2 Granular

Granular [9] is a cloud-based software that tracks crop inventories, crop variety, work orders, and schedules in the field, particularly tracking farm profitability. Granular provides an annual subscription relative to the farm size and crop variety. Granular uses client's data to improve their software and services. AuditFarm.io, on the other hand, allows the data owner to approve the algorithms that are going to run on their private data (Excel files) to generate tasks in the task management system enabling the client to preserve data ownership. Further, AuditFarm.io includes a set of tools already used by the client to provide an integrated and customized solution. Our solution offers robust auditability features for task management, leveraging a lightweight and extensible fabric network[7].

7.3 Conservis

Conservis [6] aims to manage field activities such as applications, planting, and analyze yields. Conservis is a cloud-based software as a service (SaaS) that provides harvest management, planning, inputs management, and budgeting. Conservis collects client's data. As explained in [13], moving data from/to the cloud poses significant privacy issues and can potentially expose private data. AuditFarm.io provides auditability features for task management and helps to protect data ownership via serverless oblivious smart contracts.

[6]https://serviopalacios.medium.com/farm-management-systems-2131f87c8c19

[7]The blockchain gateway allows connecting to other blockchain technologies.

8 CONCLUSION

This paper presented an auditable and integrated Task, User, Field management system that exploits a fine-grained pricing model. Our technique utilizes a novel mix of serverless functions (Azure and Twilio), distributed ledgers, serverless oblivious smart contracts, and REST APIs to provide an auditable Task Management System with elastic scaling and a fine-grained pricing model. To the best of our knowledge, this is the first solution that leverages serverless functions and shared ledgers to provide an elastic, pay-as-you-go, and auditable task management system for the AG industry preserving data ownership. We contributed an open-source implementation of a released and production-ready software and experiments that contribute empirical evidence of our solution's feasibility.

REFERENCES

[1] Marcus Brandenburger, Christian Cachin, Rüdiger Kapitza, and Alessandro Sorniotti. 2018. Blockchain and Trusted Computing: Problems, Pitfalls, and a Solution for Hyperledger Fabric. arXiv:1805.08541 [cs.DC]

[2] Miguel Castro and Barbara Liskov. 1999. Practical Byzantine Fault Tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation* (New Orleans, Louisiana, USA) *(OSDI '99)*. USENIX Association, Berkeley, CA, USA, 173–186. http://dl.acm.org/citation.cfm?id=296806.296824

[3] Chai NPM Library [n.d.]. https://www.chaijs.com/. [Online; accessed March 16, 2021].

[4] ClickUp 2021. https://www.clickup.com. [Online; accessed December 14, 2020].

[5] ClickUp API 2021. https://clickup.com/api. [Online; accessed December 16, 2020].

[6] Conservis 2021. https://conservis.ag/. [Online; accessed March 16, 2021].

[7] Farmbrite 2021. https://www.farmbrite.com/. [Online; accessed March 16, 2021].

[8] Farmlogics 2021. https://farmlogics.com/home. [Online; accessed March 16, 2021].

[9] Granular 2021. https://granular.ag/. [Online; accessed March 16, 2021].

[10] Microsoft Dynamics 2021. https://dynamics.folio3.com/. [Online; accessed March 16, 2021].

[11] Mocha NPM Library 2021. https://mochajs.org/. [Online; accessed March 16, 2021].

[12] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system". http://bitcoin.org/bitcoin.pdf.

[13] Servio Palacios. 2020. *Auditable Computations on (Un)Encrypted Graph-Structured Data.* Ph.D. Dissertation. Purdue University. https://doi.org/10.25394/PGS.12721169.v1 https://hammer.purdue.edu/articles/thesis/Auditable_Computations_on_Un_Encrypted_Graph-Structured_Data/12721169.

[14] Servio Palacios, Aaron Ault, James Krogmeier, and Bharat Bhargava. Forthcoming. AGAPECert: An Auditable, Generalized, Automated, and Privacy-Enabling Certification Framework with Oblivious Smart Contracts. (Forthcoming). Submitted.

[15] Twilio 2021. SendGrid Inbound Parse. https://sendgrid.com/docs/for-developers/parsing-email/setting-up-the-inbound-parse-webhook/. [Online; accessed January 10, 2021].

[16] Twilio Functions 2021. https://www.twilio.com/docs/runtime/functions. [Online; accessed December 20, 2020].