Design and Implementation of Blockchain-Based Office Attendance System

Yustus Eko Oktian

Department of Information Systems, Universitas Ciputra Surabaya, East Java, Indonesia
Email: yustus.oktian@ciputra.ac.id

(Received: 23 Feb 2024, revised: 13 Mar 2024, accepted: 14 Mar 2024)

Abstract

The office attendance system has shifted from using physical forms to digital inputs to minimize data errors and data loss when taking attendance. Unfortunately, digital systems generally still use traditional databases where the admin's role is crucial, and there is potential for fraud (e.g., admitting attendance of a non-attending person or manipulating a targeted person’s log due to personal grudges, competition, or other reasons) if the admin is dishonest. In this paper, we propose Absenin, a blockchain-based office attendance system, which replaces the role of traditional databases with blockchain and smart contracts to make it secure from malicious admins and fair for other participants. We create an Attendance Smart Contract that will run on the Ethereum blockchain. Admins and employees will interact with this smart contract to carry out attendance system operations. Absenin is also designed to have real-time attendance data, but the attendance machine does not need to be connected to the Internet, which is a unique feature of our system that no previous works have attempted. Despite using blockchain and smart contracts, our evaluation results show that Absenin is able to produce relatively small processing delays, and gas usage on the blockchain is still far below the gas limit of the Ethereum mainnet. Therefore, we can assure that the system is feasible and can be applied to organizations with a scale of thousands, tens, or hundreds of thousands of employees.

Keywords: Blockchain, Data Security, Attendance System, Information System

I. INTRODUCTION

With the advancement of information technology, office attendance systems are transitioning from physical to digital. Employees no longer need to fill out attendance forms in written physical forms. Instead, those forms can now be filled out digitally [1]. Digital data allows the attendance system to be represented in various forms, such as using Bluetooth signals [2], attaching RFID cards [3], scanning barcodes [4] or QR codes [5], using GPS location or cellular signals [6], and through face recognition [7]. This data is typically stored in a database, and the attendance results are generally displayed back to the user in real-time after users complete their attendance.

Unfortunately, the use of traditional databases still has shortcomings where the role of an admin is too powerful for the organization. Specifically, with the admin role, we can easily modify attendance data from the system. If the admin is malicious (or the admin account falls into the hands of malicious individuals), the admin can edit attendance data to benefit certain parties (e.g., superior officials in the organization) or harm others (e.g., employees that the admin dislikes or is currently arguing with). In addition, previous digital attendance systems [1,2,3,6,7] also have other problems, such as attendance machines that require large resources and are required to be connected to the Internet 24/7.

Driven by the motivation to solve the previously mentioned issues, in this paper, we propose a blockchain-based digital attendance system called Absenin. The use of blockchain (the technology behind the success of Bitcoin [8]) is utilized to replace the role of traditional databases so that we can improve the security and credibility of data storage in the system. More precisely, smart contracts [9] will be developed to store user attendance data so that all attendance transactions can be tracked and audited properly. This way, malicious edits can be detected easily. We utilize Merkle Root [10] to create attendance codes that will later be locked in smart contracts, similar to the way commitment hash works in blockchain [11]. Users can input the attendance code into the smart contract, and the system will verify the code and record the time when the user checked-in/out. To our knowledge, there are still few papers that utilize blockchain for attendance systems, and this paper can serve as an example to expand the development of blockchain in Indonesia.

We organize the rest of this paper as follows. Section 2 delves deeper into our proposed attendance system, starting from the motivation, design, architecture, and workflow of the system. In Section 3, we discuss the implementation process and evaluation of our system. We then investigate similar attendance systems and analyze the comparison between our system and related works in Section 4. Finally, we conclude in Section 5.

II. SYSTEM MODEL

This section outlines the inner workings of our attendance system. We begin the discussion with the problem statement that motivated us to create our proposal. We then elaborate on the design, architecture, and workflow of the attendance system to provide a deeper understanding of our system.

A. Problem Statement

Digital attendance systems can potentially reduce or even eliminate input errors or the possibility of data loss that exist in the physical system. However, conventional digital attendance systems still use traditional databases, where the role of an admin has absolute dominance in the system. For example, an admin can manipulate attendance data in the system. If the admin is malicious (or the admin's password is stolen by certain parties), then attendance data can be changed to benefit or harm certain parties. Therefore, to maintain security and fairness for all office employees, a high-integrity attendance system is heavily needed.

In addition, digital attendance systems must be built as real-time systems, where users must be able to know whether they have successfully taken attendance logs or not. They also need to know at what time they are recorded as attending/leaving the office in the system. To support real-time behavior, attendance machines are generally made to be connected to the Internet 24/7. However, this way, the machine actually becomes vulnerable to cyber-attacks, where attackers can further infiltrate the organization network through the infected attendance machines. The reliance on the Internet also means that the machine will be unavailable when problems occur in the company such as sudden power outages or Internet service downtime. Thus, a challenge arises in creating a more robust real-time attendance system that does not burden the attendance machine. The attendance machine must have low-end hardware specifications and does not even need to be connected to the Internet.

B. Features

Based on the problems mentioned earlier, we have developed an attendance system with the following features:

- **The attendance machine does not need to be connected to the Internet**. If the computer network in the given office is having problems, the attendance system can still operate normally as long as users can use another way, such as cellular data (not through the office network), to connect to the Internet.
- **The attendance machine does not require very high machine specifications**. The machine does not need to be able to process thousands of requests per second (rps) because the process of creating attendance codes is not done on-demand but can be executed in advance (pre-demand).
- **Attendance codes have an expiration period**. We can set the active period of an attendance code. Users cannot use the same attendance code to log their attending/leaving office logs. Similarly, the code for one day is not valid for the following day.
- **All attendance data is securely stored with high integrity in the blockchain**. If an audit process is required, then auditors can easily check the validity of data. We cannot create fake attendance data because the source of the attendance code is accompanied by a digital signature from the attendance machine. In addition, users also include their digital signatures when inputting their attendance log. This way, an admin cannot maliciously edit the log without being detected.

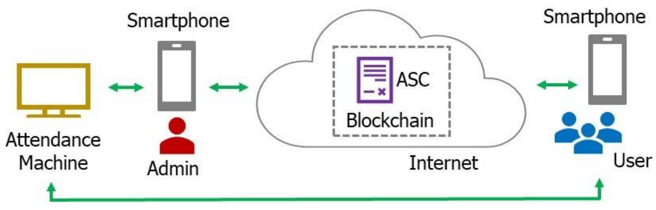


Figure 1. System Architecture for Absenin, Which Includes Admin, User, Attendance Machine, Smartphones, Blockchain, and Attendance Smart Contract (ASC).

C. System Architecture

Figure 1 illustrates the architecture of Absenin. Several actors and entities involved in our system are as follows:

- **Admin**: Individuals appointed by the organization to manage the information system in the office. Ideally, they are honest and trustworthy individuals who can manage the attendance machine effectively and refrain from malicious actions, such as distributing attendance codes out-of-band to certain parties.
- **User**: Employees from a given organization. For the sake of simplicity, we assume that no specific roles are assigned individually to each office employee, so superiors and subordinates are treated equally in our system.
- **Attendance machine**: Machines placed at various locations in the office are used to display attendance codes to users.
- **Mobile apps**: Applications used by users to input the attendance code displayed on the attendance machine for attendance purposes. Attendance data will be directly sent to blockchain nodes from the application.
- **Blockchain network**: A peer-to-peer network managed by blockchain nodes. In this network, a smart contract called Attendance Smart Contract (ASC) is used as the root-of-trust, where all actors and entities in our system highly trust this smart contract.

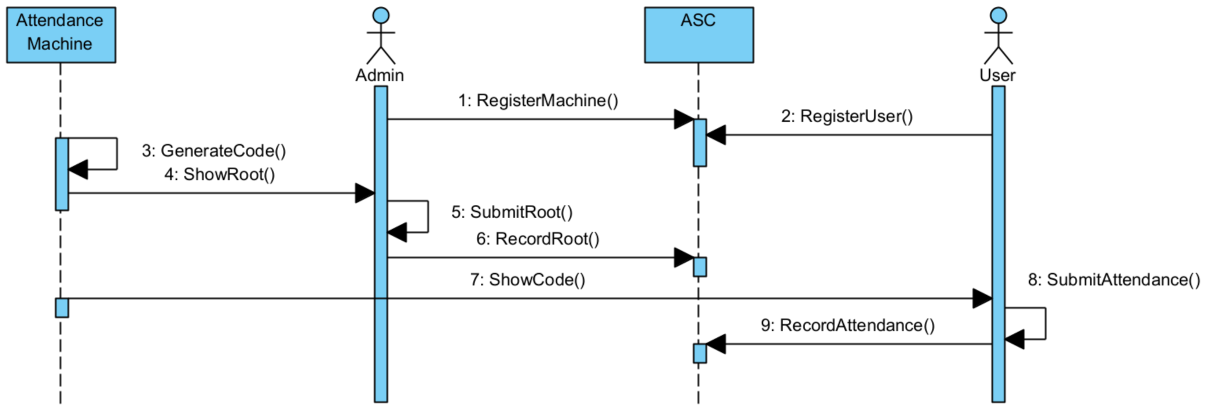


Figure 2. Workflow of Absenin, Which Comprises of Entity Registration, Attendance Code Creation, and the Usage of Attendance Code to Record Attendance in Our System.

Table 1. The Notation List That We Used in This Paper.

Notation	Description
SK, PK	Private Key, Public Key
a, u, m	Admin, User, Attendance Machine
$c, t, \hat{t}$	Attendance Code, Attendance Time, List of Attendance Logs
$r, \rho, \hat{\rho}$	Merkle Root, Merkle Path, List of Merkle Path for a single Merkle Root
t_{day}, t_{now}	Start time in a given day (in epoch timestamp), current time (in epoch timestamp)
$H()$	Cryptographic hash function (for example, using SHA-256 algorithm)
$X \parallel Y$	Concatenation of byte X and Y

Algorithm 1. Attendance Smart Contract (ASC)

```
1.  initiate $\hat{U}, \hat{M}, \hat{R}, \hat{C} = \emptyset$ // 1D array/dictionary
2.  initiate $\hat{T} = \emptyset$ // 2D array/dictionary
3.  initiate $\hat{A}$, where $\hat{A} = \{PK_{a_1}, PK_{a_2}, \dots, PK_{a_A}\}$; $A$ is
      the total number of admin

4.  procedure RegisterUser($PK_u$):
5.    store $PK_u$ in $\hat{U}$

6.  procedure RegisterMachine($PK_a, PK_m$):
7.    revert if $PK_a \notin \hat{A}$
8.    store $PK_m$ in $\hat{M}$

9.  procedure RecordRoot($PK_a, PK_m, r$):
10.   revert if $PK_a \notin \hat{A}$
11.   revert if $PK_m \notin \hat{M}$
12.   store $r$ in $\hat{R}$

13. procedure RecordAttendance($PK_u, PK_m, c, \rho$):
14.   revert if $PK_u \notin \hat{U}$
15.   revert if $PK_m \notin \hat{M}$
16.   revert if $c \in \hat{C}$
17.   make merkle root $r'$ out of attendance code $c$
        and merkle path $\rho$
18.   revert if $r' \notin \hat{R}$
19.   store $c$ in $\hat{C}$
20.   $t_{day} = t_{now} - (t_{now} \bmod 86400)$ // 86400 =
        seconds in a day
21.   revert if $t_{now} > t_{day} + 86400$
22.   get $\hat{t}$ from $\hat{T}$ using key $t_{day}$ and $PK_u$
23.   add $t$, where $t = t_{now}$, to $\hat{t}$

24. function GetArriveTime($PK_u, t_{day}$):
25.   get $\hat{t}$ from $\hat{T}$ using key $t_{day}$ and $PK_u$
26.   return smallest $t$ from $\hat{t}$

27. function GetLeaveTime($PK_u, t_{day}$):
28.   get $\hat{t}$ from $\hat{T}$ using key $t_{day}$ and $PK_u$
29.   return largest $t$ from $\hat{t}$
```

D. Workflow

Our system can be divided into six major steps: (i) entity registration process, (ii) attendance code creation process, (iii) attendance code display, (iv) attendance code submission, (v) attendance code verification, and (vi) storing of attendance codes. The system workflow is summarized in Figure 2. Meanwhile, Table 1 shows the notations that we use for the rest of this paper.

Registration Process: Users who wants to use our attendance system must first register their public key in the ASC through the `RegisterUser()` method. Similarly, admin also needs to register each attendance machine to the ASC using the `RegisterMachine()` method. In this case, we assume that the admin's public key has been initially created and registered in the ASC when ASC is deployed in the blockchain network. This registration process is crucial because we implement access control system within the ASC, where several ASC functions can only be executed by the admin. The complete operation of the ASC can be seen in Algorithm 1.

Code Generation Process: The attendance machine first generates an attendance code through the `GenerateCode()` function by: (i) creating a random code, then hashing the code combined with the public key of the machine. The machine will generate several codes at once for a certain period (e.g., one hour, one day, or one month). The generated codes will be arranged to form a Merkle Tree, and the Merkle Root result will be displayed to the admin through the `ShowRoot()`

function and submitted by the admin to the ASC through the `SubmitRoot()` and `RecordRoot()` functions. Details of how the attendance machine works can be seen in Algorithm 2, and for the admin mobile apps in Algorithm 3.

Attendance Code Display Process: The attendance machine will display the attendance code through the `ShowCode()` function sequentially from one code to another. One code will only appear for a few seconds (e.g., 3 seconds) and one code can only be used for one user. The code also only appear once in one generation period. For example, if we generate 1000 codes to be displayed for one hour, code A will only appear once in that 1-hour duration; code A cannot (and should not) appear twice. The system is designed this way because the attendance machine is offline and cannot detect which codes have been used by users for attendance. Therefore, the machine only displays the code for a very short duration with the hope that the code can only be seen and used by one user.

Algorithm 2. Pseudocode for Attendance Machine

```
1.  initiate $SK_m, PK_m, r, \hat{\rho}$
2.  initiate $\hat{C} = \theta$ // 1D array/dictionary
3.  procedure GenerateCode($n$): // $n$ is the number
      of codes to make
4.    i = 0
5.    while $i < n$:
6.      $id = CreateRandomID()$
7.      $c = H(PK_m \parallel id)$
8.      store $c$ in $\hat{C}$
9.      i++
10.   end while
11.   $(r, \hat{\rho}) = CreateMerkleRoot(\hat{C})$
12. procedure ShowCode():
13.   foreach $c$ in $\hat{C}$:
14.     get $\rho$ for $c$ from $\hat{\rho}$
15.     display $(PK_m, c, \rho)$ for 3 seconds
16.   end foreach
17. procedure ShowRoot():
18.   display $(PK_m$, r)
```

Algorithm 3. Pseudocode for Mobile Apps in Admin

```
1.  initiate $SK_a, PK_a$
2.  procedure SubmitRoot():
3.    get $(PK_m, r)$ via out-of-band from ShowRoot()
        in Algorithm 2
4.    call RecordRoot($PK_a, PK_m, r$) in ASC
```

Algorithm 4. Pseudocode for Mobile Apps in User

```
1.  initiate $SK_u, PK_u$
2.  procedure SubmitAttendance():
3.    get $(PK_m, c, \rho)$ via out-of-band from
         ShowCode() in Algorithm 2
4.    call RecordAttendance($PK_u, PK_m, c, \rho$) in ASC
```

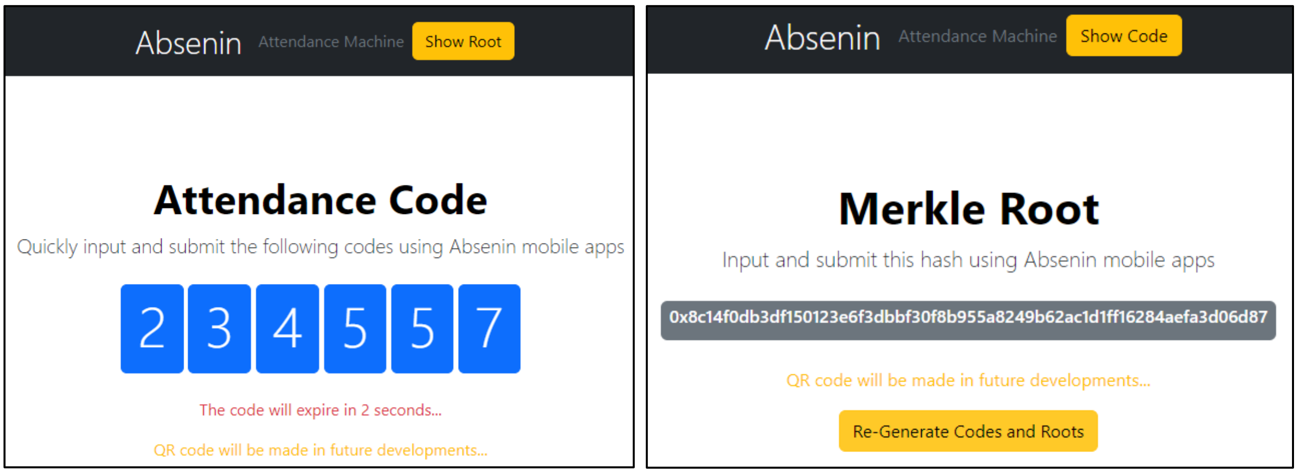
Attendance Code Collection Process: If there are two or more users who are found to be using the same code, the "first come, first serve" rule will apply. The user who first inputs the code into the ASC through the `SubmitAttendance()` and `RecordAttendance()` functions will be considered valid, while subsequent users using the same code will fail. Failed users must wait for the next code to appear to successfully checked-in. The process of inputting the code into the blockchain is online and real-time, so users will receive feedback from the system indicating whether their attendance was successful or not. Details of the user's mobile app workflow can be seen in Algorithm 4.

Attendance Code Verification Process: Users collect the attendance code, merkle root hash, merkle path, and public key from the attendance machine when checking-in through `RecordAttendance()`. After that, the ASC verifies the user's input. First, the ASC checks whether the public key of the attendance machine is already registered with the ASC. Second, the ASC ensures that this attendance code has not been used by other users. Third, the ASC creates a root hash of the attendance code and merkle path, then the ASC will compare whether the newly formed root hash is the same as the root hash originally saved during the attendance code creation process through the `RecordRoot()` function. Fourth, the ASC ensures that the code has not expired. If all verification processes are successful, the ASC assigns this attendance code to the user who called `RecordAttendance()`, so this code cannot be used by other users.

Attendance Code Storage Process: The attendance code is stored along with the time (in Epoch time format) indicating when this code was processed. This time represents when the user checked-in. The ASC will store at least two attendance inputs from the user. The earliest input in one day is the attending office time, and the latest input on the same day will be considered as the leaving office time. If a user has fewer than two inputs in one day, then the user has invalid attendance (considered as not coming to the office). To get the arrival and departure attendance, users can call the `GetArrivalTime()` and `GetLeaveTime()` functions on the ASC.

III. EXPERIMENTAL RESULTS

Absenin can be connected to any blockchain (whether permissioned or permissionless) as long as they are compatible with the Ethereum Virtual Machine (EVM) [12]. For this purpose, we utilize hardhat [13] as a blockchain emulator and development tools. Smart contract code is written using Solidity, while applications for admin, user, and attendance machines are implemented using Typescript. After the prototype was built, we analyzed the performance of our system in several aspects, including processing delay and gas usage. We tested our system using a computer with a specification of Intel Core i5-8250U CPU @ 1.60 GHz and SK Hynix DDR4 RAM @ 2400 MHz. During the evaluation process, we only used 1 CPU core and 8 GB of RAM. Figure 3 shows the user interface for our attendance machine prototype.



Attendance Code

Quickly input and submit the following codes using Absenin mobile apps

2 3 4 5 5 7

The code will expire in 2 seconds...

QR code will be made in future developments...

Merkle Root

Input and submit this hash using Absenin mobile apps

0x8c14f0db3df150123e6f3dbbf30f8b955a8249b62ac1d1ff16284aefa3d06d87

QR code will be made in future developments...

Figure 3. The User Interface in the Attendance Machine Showing the Attendance Code (Left) and the Merkle Root (Right) to be Processed by the Mobile Apps in Admin And Users.

Table 2. Processing Delay (in Milliseconds) for Methods in ASC When Processing Diverse Number of Attendance Code

Method	**Number of Attendance Code**		
	28800	**43200**	**86400**
`GenerateCode()`	264.23	371.92	710.89
`MerkleRoot()`	70.60	1,141.46	2,070.12
`MerkleProof()`	178,510.70	464,313.19	2,114,973.89
`SubmitRoot()`	16.63	12.33	24.71
`SubmitAttendance()`	983,767.97	1,221,684.77	2,298,884.54

A. Processing Delay

We first analyzed Absenin to identify potential bottlenecks in our system. Suppose if one attendance machine displays 1 code every 3 seconds, then in one day the machine needs to prepare 28800 attendance codes. If the code display duration is shortened to 2 or 1 second, then the machine would need 43200 and 86400 attendance codes, respectively. For this reason, we conducted a simulation evaluation to process 28800, 43200, and 86400 attendance codes to cover three different scenarios of code display duration. Specifically, we utilize the Node JS Performance module to benchmark the processing delay from core methods in Absenin as seen in Table 2.

Based on Table 2, we can conclude that the Merkle Proof creation process is the major bottleneck in the system. In this process, the attendance machine has to create proofs for all the attendance codes that have been generated. The more attendance codes that need to be generated, the greater the processing delay of the system. However, the good news is that the process of generating attendance codes, Merkle Root, and Merkle Proof only needs to be done once per day. Therefore, this process can be done outside of office hours (for example, early in the morning) so it will not disrupt the operation of the attendance machine during work hours.

Our analysis indicates that the `SubmitAttendance()` process takes 16-38 minutes. This number may seem very large, but it is important to note that this delay is for processing the total 28800, 43200, or 86400 codes. In the operational process of the attendance machine, one user only needs to enter the code twice (for check-in and check-out), and we also limit one code to be processed every 3, 2, or 1 second. Therefore, this large delay is not considered as a bottleneck. The process of recording one attendance code through the `SubmitAttendance()` function only takes about 26-34 milliseconds.

B. Gas Consumption

Every process that changes the state of the blockchain network in the EVM will be subject to gas calculation [14] and will be forced to stop and return to the original state if the process is run out of gas or exceeds the upper limit of gas consumption. Currently, the maximum limit is 30 million gas per block. The limit is enforced to avoid the possibility of an endless loop, where a blockchain node run processes continuously and wastes all their resources. Therefore, a good and efficient blockchain system is one that uses gas as small as possible and stays far below the maximum limit. Table 3 shows the gas consumption of ASC functions, which is recorded using the hardhat-gas-reporter module.

Table 3. Gas Usage in ASC

Method	Gas	%Limit
`RegisterUser()`	23,683	0.08
`RegisterMachine()`	26,711	0.09
`RecordRoot()`	29,811	0.10
`RecordAttendance()`	76,449	0.25

Based on our evaluation results, the registration process for users, attendance machines, and the submission process for Merkle Roots require relatively small amounts of gas. These processes also only need to be done once, so they do not burden the system too much. However, the attendance process performed by users through `RecordAttendance()` is a different case. Each user needs to call this function at least twice a day. Therefore, this function may result in significant gas usage and become the bottleneck in Absenin.

With a gas consumption of 76,499, we can only include about 392 attendance codes per block. If a block is created every 3 seconds, then the system can only record 130 attendance codes per second. This value is still sufficient for recording attendance from offices with thousands, tens of thousands, or even hundreds of thousands of employees. If the performance is still considered insufficient, we can reduce the block creation period from 3 seconds to 1 second so that now every second we can process 392 attendance codes.

It should be noted that we only display write methods in Table 3, while read-only methods (e.g., `GetArriveTime()`, `GetLeaveTime()`) are not displayed because they do not require gas when executed in the EVM.

C. Weakness

In the last part of this section, we discuss several weaknesses of Absenin that we have identified. We also elaborate on suggestions and strategies to eliminate or mitigate these weaknesses.

Long Queues: An attendance code can only be used for every few seconds (one code per three seconds in our example). In cases where many employees suddenly arrive at the same time, there will be a queue. If 20 people arrive simultaneously, then the last person to arrive must wait at least one minute before they can take attendance.

There are two solutions for this case. First, we can reduce the duration of the code display to two or one second to reduce the waiting time. However, this requires the attendance machine to generate more codes compared to a three-second duration. In result, the machine needs more computing and storage resources. Alternatively, we can also increase the number of attendance machines at points where crowds may occur to minimize queues. It should be noted that, depending on the situation in the office, the occurrence of 20 people arriving together may be considered a rare event. So, a little bit of queues should be acceptable.

Attendance Codes Can Still Be Misused: Users can learn how our attendance system works through trial and error. If users find out that unused codes that have shown from the attendance machine screen can still be used, then users can memorize or record several attendance codes and then distribute them to other users. This way, users can attend remotely.

To address these issues, a near-field authentication process should be used (for example, using Bluetooth Low Energy, RFID), where the user's smartphone must be close enough to the attendance machine during check-in for it to be considered valid. However, this approach may result in a terrible user experience because the authentication process is not convenient. For example, Bluetooth connections are susceptible to interferences, while RFID requires users to bring additional devices (RFID tags) to work, aside from their smartphones.

Admin Still Has to Configure Attendance Manually: In our system, the admin must setup the attendance machine to generate attendance codes and then manually add the Merkle Root of the codes to the ASC. If the admin forgets to do this, the attendance system cannot be used. The solution to this issue is the admin can create and register multiple Merkle Roots for future periods automatically through a scheduling process.

Table 4. Feature Comparison of Absenin With Related Works Regarding Attendance Method and Whether They Have Blockchain (BC.), Offline Attendance Machine (Off.), Real-Time System (RT.), Integrity (Int.), and Expiry Time / Time Limit for Attendance Code (Exp.).

Ref.	Attendance Method	BC.	Off.	R.T.	Int.	Exp.
[2]	Bluetooth	✗	✗	✓	✗	✗
[7]	Haar Cascade Classifier	✗	✗	✓	✗	✗
[4]	Fingerprint, Barcode	✗	✗	✗	✗	✗
[5]	QR Code	✗	✗	✗	✗	✗
[3]	RFID	✗	✗	✓	✗	✓
[6]	QR Code, LBS	✗	✗	✓	✗	✗
[15]	Barcode	✓	✗	✓	✓	✗
[1]	Attendance Form	✓	✗	✓	✓	✗
Ours	Attendance Code	✓	✓	✓	✓	✓

IV. RELATED WORK

In this chapter, we present previous studies on the topic of "information systems for tracking user attendance in an organization." It should be noted that we only filter papers that have implementation in Indonesia, by only including papers written in Bahasa Indonesia for comparison. Table 4 summarize the comparison of our system with related works.

P. R. Setiawan [2] created an Android-based attendance system. The author developed this application to address the common problem of "signing in an absent person" among students to meet the minimum attendance requirement.

Munawir et al. [7] developed an attendance system utilizing face detection with the Haar Cascade Classifier. The system is capable of detecting one face or multiple faces simultaneously. Handayani et al. [4] analyzed the attendance system of PT. Ambassador Garmindo Sukoharjo to identify the strengths and weaknesses of the system, as well as to identify zero-days problems.

Asiking et al. [5] developed an attendance system using Quick Response Code technology equipped with Secure Hash Algorithm (SHA) and BCRYPT algorithm to secure the attendance process performed through an Android smartphone. Zakaria et al. [3] utilized Radio Frequency Identification (RFID) for student attendance. Data from RFID will be stored in a MySQL database for easy and centralized attendance data log recording. Sikumbang et al. [6] used Location Based Service (LBS) to take attendance log at the Bandung City Statistics Center when performing official duties outside the office. The user's location will be monitored through an Android application as proofs of attending office.

For blockchain-based attendance systems, Hidayat et al. [15] developed an online attendance system based on granted data validity and barcode scanning. The authors used blockchain to secure attendance data stored in the system. Wijaya et al. [1] also used blockchain to secure the attendance system at the Lembaga Pengembangan Komputerisasi Mandiri berbasis Virtual (VM LePKom) Universitas Gunadarma. The authors revamped the original system, which still used traditional databases, to use blockchain so that all attendance data could be securely stored.

The majority of attendance systems mentioned previously (cf. Table 4) already use a real-time system in which users can instantly check whether their attendance process was successful or not. However, previous works require the attendance machine to be connected to the Internet for them to provide real-time feedback. In contrast, our attendance machine is not required to be connected to the Internet at all. Furthermore, most attendance systems do not specify whether their attendance process has a time limit or not, so there is a possibility that attendance codes (such as QR codes, barcodes, fingerprints, Bluetooth signals) can be used multiple times and abused by users. Additionally, among all the papers discussed, only two of them utilize blockchain. This indicates that the discussion on attendance systems in Indonesia is still mostly focused on conventional methods using a database system (non-blockchain). Therefore, this paper can contribute as an example of implementing blockchain in the field of attendance systems to promote the use of blockchain in Indonesia.

V. CONCLUSION

In this paper, we have proposed Absenin, an office attendance system that utilizes blockchain technology. Our system offers novel features such as an attendance machine that does not require an internet connection but still provides a real-time attendance process for users. Additionally, the use of blockchain in our system provides better security and integrity properties compared to traditional attendance systems that utilize conventional databases. Based on the evaluation results, we conclude that the system is feasible and can run correctly as initially designed. Processing delay is also relatively small, and the average gas usage is below the limit. The major bottleneck in our system is the generation of Merkle Proof and the submission of attendance code, which should become the point of interest during deployment.

For future work, we plan to implement the system on commercial hardware to test the performance of using small-capacity machines such as Raspberry Pi 3 as attendance machines. Additionally, attendance codes can be transformed into QR codes to simplify the attendance process. Users only need to scan the QR code displayed on the attendance machine. In a broader use case, our proposal can also be used to log anything from the physical world to the digital world. For example, in the supply chain, Absenin can be tweaked to log one of the child processes in the entire supply chain area. This way, we can have a credible log and know how much process is needed from child A to child B, and so on.

REFERENCES

[1] I. Wijaya, E. Haryatmi, and A. B. Kurniawan, ‘Implementasi Teknologi Blockchain pada Sistem Presensi Staff VM LePKom Berbasis Web’, *Jurnal Nasional Informatika dan Teknologi Jaringan*, vol. 5, no. 1, pp. 162–169, 2020.

[2] P. R. Setiawan, ‘Aplikasi Absensi Online Berbasis Android’, *IT Journal Research and Development*, vol. 5, no. 1, pp. 63–71, 2020.

[3] A. Zakaria and A. Prihantara, ‘Pemanfaatan Radio Frequency Identification Mifare RC522 dan Arduino Sebagai Media Validasi Kehadiran Mahasiswa’, *Jurnal Infotekmesin*, vol. 11, no. 01, 2020.

[4] S. Handayani, P. Ninghardjanti, and A. Subarno, ‘Pengelolaan Sistem Informasi Presensi Di Pt Ambassador Garmindo Sukoharjo’, *JIKAP (Jurnal Informasi Dan Komunikasi Administrasi Perkantoran)*, vol. 4, no. 4, pp. 37–52.

[5] A. Asiking, I. S. K. Idris, and Others, ‘Quick Response Code Absensi Guru Menggunakan Secure Hashing Algorithm (SHA)’, *Jurnal Tecnoscienza*, vol. 6, no. 2, pp. 332–346, 2022.

[6] M. A. R. Sikumbang, R. Habibi, and S. F. Pane, ‘Sistem informasi absensi pegawai menggunakan metode RAD dan metode LBS pada koordinat absensi’, *Jurnal Media Informatika Budidarma*, vol. 4, no. 1, pp. 59–64, 2020.

[7] M. Munawir, L. Fitria, and M. Hermansyah, ‘Implementasi Face Recognition pada Absensi Kehadiran Mahasiswa Menggunakan Metode Haar Cascade Classifier’, *InfoTekJar: Jurnal Nasional Informatika dan Teknologi Jaringan*, vol. 4, no. 2, pp. 314–320, 2020.

[8] S. Nakamoto, ‘Bitcoin: A peer-to-peer electronic cash system’, *Decentralized business review*, 2008.

[9] A. M. Antonopoulos and G. Wood, *Mastering ethereum: building smart contracts and dapps*. O'reilly Media, 2018.

[10] K. S. Garewal and K. S. Garewal, 'Merkle trees', *Practical Blockchains and Cryptocurrencies: Speed Up Your Application Development Process and Develop Distributed Applications with Confidence*, pp. 137–148, 2020.

[11] Y. Manevich and A. Akavia, 'Cross chain atomic swaps in the absence of time via attribute verifiable timed commitments', *in 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, 2022, pp. 606–625.

[12] G. Wood and Others, 'Ethereum: A secure decentralised generalised transaction ledger', *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[13] S. M. Jain, 'Hardhat', *in A Brief Introduction to Web3: Decentralized Web Fundamentals for App Development*, Springer, 2022, pp. 167–179.

[14] M. Lepcha. Gas Fee (Ethereum). *Techopedia*. [Online] Available: https://www.techopedia.com/definition/gas-fee-ethereum

[15] M. M. Hidayat, A. A. Mubarrok, B. P. B. Utomo, and M. I. Zacharia, 'Perancangan Sistem Presensi Online Berbasis Granted Validitas Data', *Jurnal Informatika dan Teknologi (INTECH)*, vol. 4, no. 1, pp. 23–27, 2023.