IOTA-MSS: A Pay-per-Play Music Streaming System based on IOTA

Daniel Melero Martinez
Semantics, Cybersecurity & Services
University of Twente
Enschede, Netherlands
d.meleromartinez@student.utwente.nl

Mohammed Elhajj
Semantics, Cybersecurity & Services
University of Twente
Enschede, Netherlands
m.elhajj@utwente.nl/0000-0002-4022-9999

2023 9th International Conference on Computer and Communication Engineering (ICCCE) | 979-8-3503-2521-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCCE58854.2023.10246055

***Abstract*—In recent years, music streaming has emerged as the primary method for users to enjoy their favorite songs. Music Streaming Services (MSS) have risen to become influential entities in the music industry, exemplified by Spotify's impressive 406 million monthly active users by the end of 2021. However, many artists express concerns that the current MSS model fails to adequately compensate them for their music, raising questions about fairness. To tackle this issue, this paper proposes the implementation of IOTA-MSS, a Pay-per-Play Music Streaming System that leverages the secure and scalable IOTA distributed ledger technology (DLT). By harnessing the capabilities of IOTA's DLT, IOTA-MSS offers a platform that enables seamless micro-transactions. This innovative system empowers users to play and distribute music in real-time while ensuring that rights holders receive corresponding payments. Through the utilization of IOTA's DLT, IOTA-MSS addresses the challenges faced by artists, providing a more equitable solution for music compensation. By adopting a Pay-per-Play model, this system aligns artist remuneration with the actual usage and popularity of their music. Furthermore, IOTA-MSS enhances the user experience, offering a secure and efficient platform for music enthusiasts to engage with their favorite tunes.**

***Index Terms*—Music Streaming, DLT, IOTA, Security, Scalablity.**

I. Introduction

The rapid advancement of the Internet of Things (IoT) has revolutionized the way we interact with technology, enabling billions of interconnected devices to gather and exchange data [1]. As IoT continues to expand, ensuring secure and efficient transactions among these devices becomes increasingly crucial [2], [3]. This is where Distributed Ledger Technology (DLT) comes into play, and one prominent DLT platform specifically designed for the IoT ecosystem is IOTA [4]. By leveraging IOTA's innovative approach to decentralized ledger technology, IoT devices can securely [5] exchange data and conduct transactions in a trustless and scalable manner. The integration of IoT with IOTA opens up new possibilities for enhancing data integrity, enabling micro-payments, and facilitating seamless interoperability among IoT devices, leading to a more secure, efficient, and interconnected IoT landscape. The synergy between IoT and IOTA opens up opportunities for enhanced security, improved data integrity, and streamlined transactions in various IoT use cases, including supply chain management, smart cities, autonomous vehicles, music streaming, etc. Music Streaming Services (MSS) have dominated the recorded music industry's revenue since 2017, with increasing numbers of MSS expected in the future. Spotify, the leading platform with 406 million monthly active users in 2021, generates revenue mainly from premium subscriptions and adopts a service-centric model for artist royalties based on total streams. However, concerns are raised about the fairness of this model for musicians and their audiences [6], [7].

A. User's privacy

Most MSS offer free services to attract users, but increasing operational costs and difficulty in converting users to paying subscribers pose challenges for platforms. To address this, streaming platforms have expanded beyond music services to gather and exchange user data. For instance, Spotify modified its privacy policies in 2015 to collect personal information from users' devices, highlighting privacy concerns in the current service-centric model of MSS [8].

B. Musician's compensation

The current system of music streaming platforms is criticized for not benefiting music rights owners and creating an unfair environment in the industry, primarily due to royalty arrangements, playlists, limited access to critical data, and the dominant negotiating power of major labels [9]. This anti-competitive behavior particularly disadvantages independent musicians who lack the resources of major labels. Musicians face challenges in increasing their revenue as they have limited control over negotiating contracts or influencing the overall revenue-sharing practices of MSS [10]. This lack of control exemplifies how musicians have limited agency in the distribution and monetization of their music.

C. User's inability to support musicians

The lack of a sense of control can also be said for the users of these platforms as they are required to pay a monthly fee which does not correlate with their use of the platform. "In the present service-centric system, even if that user never played a single recording by superstars such as Drake or Taylor Swift, a proportion of that user's subscription fee would go to the owners of rights to those superstars' recordings and underlying compositions" [10]. This means that even if musicians have

979-8-3503-2521-8/23/$31.00 ©2023 IEEE

an established audience willing to pay for their music, they might still be unable to support themselves only through their published material.

D. Research Questions

This paper proposes a novel decentralised music streaming application based on a pay-per-play model. The solution is implemented using the distributed ledger IOTA which provides a secure and scalable platform to perform micro-transactions. The solution allows music rights holders to decide the value of their music, and users to only pay for the music they listen to. The goal of this paper is to provide a thorough response to the following question:

How can IOTA benefit both the supply and demand sides of the music industry?

To reach a detailed conclusion the paper first responds to the following questions:

- *How can IOTA smart contracts be used to perform rapid high-throughput transactions?*
- *How can data be transmitted in parallel with its corresponding payment?*
- *How can IOTA be used to ensure users' privacy?*

The structure of this paper is as follows, Section II covers a literature review of proposed solutions to similar problems. Section III provides a detailed explanation of the solution proposed by this paper and Section IV collects the results of the analyses run on the implementation. Finally, Section V concludes the paper and discusses some ideas for future research to improve the proposed solution's security and reliability.

II. Related Works

There exist a few proposed decentralized applications aimed at music streaming in the literature. However, none of these existing solutions uses the IOTA distributed ledger.

The primary work referenced in this context proposes a pay-per-play music application utilizing the Ethereum blockchain for payment transactions, and IPFS for music distribution [11]. The blockchain records transactions and publishes smart contracts for managing recipients. The model allows users to listen to songs for free, with the option to tip artists. In this model, 25% of earnings from each mined blockchain block are distributed to artists per stream, while the remainder goes to miners. Songs are stored in IPFS nodes maintained by the artists. Ethereum Smart Contracts enable real-time rewards for music right owners based on stream count. However, Ethereum's scalability limitations prevent it from replacing existing streaming platforms, and the absence of direct payment per stream could leave the system vulnerable to click-fraud issues observed in major platforms [12].

While no specific solution for music streaming using IOTA has been proposed, research suggests the use of IOTA for data marketplaces. These marketplaces involve IoT device owners with sensor data and users interested in accessing that data. Although data marketplaces don't involve streaming, they share similarities with music streaming platforms, as the transferred data can also include audio files.

In [13], a decentralized energy trading marketplace using IOTA 2.0 Smart Contracts is proposed. IOTA Smart Contracts (ISC) enable uniform double-auctions for trading excess energy in interconnected microgrids. Each microgrid has a dedicated ISC implemented on a separate wasp chain, where buyers and sellers can place bids on energy prices. Comparative testing between IOTA and Ethereum shows IOTA's cost to interact with ISC is considerably lower due to fixed fees and the dynamic Proof of Work in IOTA 2.0 consumes less energy.

Similarly, in [14], a privacy-focused data marketplace utilizing IOTA 2.0 Smart Contracts is discussed. The marketplace considers decentralized data storage (e.g., IPFS or Swarm) to store encrypted data, ensuring indirect interaction between buyers and sellers. A distributed certificate authority is employed for data authenticity. The application's logic is implemented on three different L2 chains: the buyers' Smart Contract, the sellers' Smart Contract, and the broker's Smart Contract. The broker, running in a permissioned environment, acts as an intermediary between buyers and sellers, enhancing privacy by separating events and making data tracing challenging.

In [15], IOTA 1.5 is utilized to create the Streaming Data Payment Protocol (SDPP), enabling micropayments for streaming IoT data. The protocol employs a standard TCP connection for data transmission and IOTA value transfers for payment and record-keeping. However, the implementation was deemed slow due to the immaturity of the IOTA tangle in 2018. The researchers use Masked Authenticated Messaging (MAM) for the recording medium since IOTA 1.0 lacks Smart Contracts.

The same research team develops a Decentralized Data Marketplace in [16], utilizing the SDPP protocol. IOTA serves as the payment channel, while IPFS and Ethereum Smart Contracts store product offers and transaction records. The resulting prototype is a web application. Since IOTA 1.0 lacks Smart Contracts, the Ethereum blockchain [17] is used for the main smart contract, but it incurs fees and requires distributed file storage to mitigate costs.

In [18], a P2P file-sharing system integrating BitTorrent with IOTA is proposed. The protocol operates basic BitTorrent functions on an IOTA distributed ledger, providing enhanced security. The protocol reduces unnecessary transaction searches, improving seeding and heartbeat processes. This project solely relies on IOTA 1.0, employing personalized IOTA nodes and MAM transactions. While it focuses on facilitating BitTorrent, the tangle tracks shared files and seeders without utilizing IOTA's microtransactions. The researchers suggest leveraging IOTA 2.0 smart contracts to transform the system into a P2P file trading system, enabling BitTorrent functionalities within the IOTA network.

Several decentralized marketplaces not based on IOTA have

been proposed. A notable one is discussed in [19], where different blockchains are evaluated. IOTA is deemed unsuitable due to centralization concerns and storage limitations in its 2018 version. The researchers aim for an ”always-on” marketplace, requiring smart contracts that were absent in IOTA’s initial version. Instead, they suggest using Ethereum Smart Contracts and Swarm, a decentralized storage service similar to torrents. The paper provides a detailed action flow between vendors and customers, along with the smart contract code. All application logic is contained in one smart contract, and both vendors and customers interact with it for trades. Data is encrypted, stored in Swarm, and registered through the smart contract. Customers request data from the smart contract, which notifies the vendor. The vendor then shares the decryption key with the customer for data retrieval. The project successfully implements a decentralized marketplace using Ethereum. However, Ethereum’s slow block creation times and transaction fees limit high volumes of microtransactions. To address this, the project employs payment channels as an off-chain solution to reduce latency and fee costs.

III. Implementation

This section will first give an overview in III-A of the different technologies used by the platform and how these interact with each other. Then, a more detailed explanation of how the user interacts with the platform is given in III-B and how transactions are managed within the smart contract is detailed in III-C. Finally, the main actions that the users can execute and how they affect the state of the smart contract are explained in more depth. In III-D how creating user accounts works, in III-E how music is uploaded to the platform and in III-F how sessions are used to listen or download music.

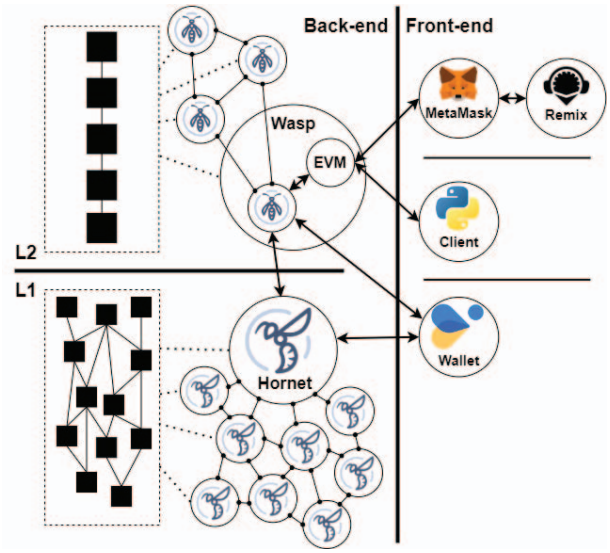
A. Platform Overview

The platform can be divided into two different groups: the back-end and the front-end. The following diagram shows the different technologies which form each group and how these interact with each other.

The technologies required to run the smart contract are depicted on the left side of Figure 1. The IOTA ledger L1, also known as the tangle, is supported by the Hornet node. The smart contract runs on the Wasp node, which constitutes the chain (L2) anchored to the L1 ledger. For testing purposes, only one instance of each node runs on the same local network, but in practice, multiple computers would ensure ledger consensus.

On the right side of Figure 1, various technologies are used by different stakeholders to interact with the smart contract and the ledgers. The administrator’s front-end employs MetaMask and Remix IDE for managing the platform, including smart contract compilation, deployment, and low-level transactions. The user’s front-end consists of the python client, which connects to the EVM using Web3.py, establishes encrypted connections with other clients, and plays received music using python-vlc. Both administrators and users utilize the wallet

Figure 1: Diagram of the proposed solution



wasp-cli to set up new chains and transfer funds between L1 and L2.

Together, the front-end and back-end technologies form the infrastructure of the proposed open-source music streaming platform. The source code for the python client, smart contract, and deployment instructions on a local network can be found in Github.

B. Platform’s Action Flow

All users interact with the platform through the python client which is a command-line interface used to perform basic actions intuitively. The following image shows the main menu of the client and therefore all the possible actions.

All users first have to deposit some funds to their chain address and create an account with their name and a description. Then, users are able to upload music, listen, download, and serve songs to other users, monitor the activity of the server, and withdraw the funds generated or unused. These individual actions modify the state of the smart contract which simply means that the data of the program is updated acting as a sort of database. The main data structures and how they are used is explained in the next sections.

C. Real-time compensation

Performing micro-transactions in the L2 chain can decrease platform efficiency and increase gas costs. To address this issue, the smart contract abstracts transactions within an accounting system.

Users transfer MIOTAs, the tangle’s currency, to their chain address. The tokens are held in the chain’s wallet, which provides the equivalent amount to the user’s address in the chain. Users can then deposit funds into their smart contract account using the payable function `deposit()`. The smart contract holds the currency, tracking the user’s balance.

When a user pays for music, a non-monetary transaction is sent to the smart contract, triggering the accounting function.

This reduces the number of transactions since only one is required, and the tokens remain in the smart contract address.

To withdraw their account balance, users can use the `withdraw(uint amount)` function, transferring their balance to their tangle address in MIOTAs. The smart contract sends the tokens to the EVM contract, initiating a transaction from the chain's wallet to the user's tangle address with the corresponding MIOTA tokens.

Therefore, actual IOTA token transactions only occur when users choose to withdraw their account balance. Users can withdraw whenever their balance increases for real-time compensation, but this is not the default to facilitate rapid high-throughput exchanges. Additional details on user accounts in this solution are provided in the following section.

D. User creation

The user structure stores necessary user information. When a user account is created, a new instance of this structure is added to the users map , mapping the user's chain address to their information. Initially, the structure contains the name, description, and "exists" set to True, with other fields set to 0, False, or empty string.

The server field can be updated to include the user's server information for music distribution. Funds can be added from the On-Chain balance using the `deposit()` function, through uploads or serving music. The balance can be reduced by withdrawing funds with `withdraw(uint amount)` or by using funds to listen to others' music.

The last field, validation, determines the ability to upload music, explained in detail in the following section.

E. Music uploading and distribution

The data structure stores necessary information about a music file. Similar to users, when a music file is uploaded, a new instance of this structure is mapped to its ID. The ID is generated as the keccak256-hash value of the music's name and the owner's address, allowing multiple files with the same name to be uploaded. The ID is added to the list of files to indicate its availability. Function `upload_music()` defines the required information for uploading a music file, including the author's address, name, price, length (in bytes), duration (in seconds), and the list of chunks (identified by keccak256-hash values).

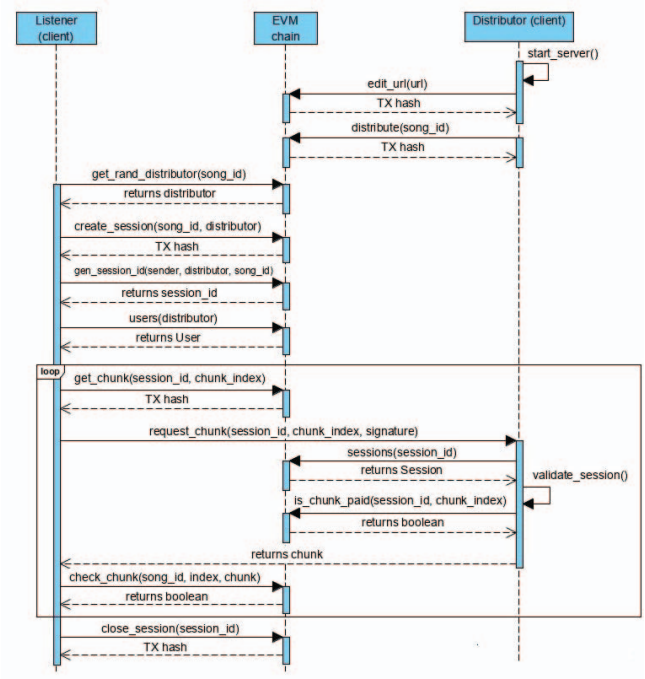
After uploading, files are not immediately available to prevent unauthorized sharing. The responsibility of determining which files can be shared is delegated to the smart contract's deployer. The owner variable allows the deployer to define users as validators using the function `manage_validators(address user)`. Validators can grant or deny validation for files using `manage_validation(bytes32 music)`.

The "valid" status indicates that a music file can be listened to and distributed. Initially, the author is set as the only distributor since they are the only one with the mp3 file. Other users can sign up to be distributors using `distribute(bytes32 music)`. To distribute a file, users need to possess the entire file and create a session to request each chunk. The process is further explained in the following section.

F. Session management

Sessions are used to manage the distribution of valid music between two users, a listener and a distributor. This process must be performed efficiently to allow for a decent user experience which is why both users should be using the python client. The sequence diagram in Figure 2 shows an overview of how both clients interact with the back-end and with each other.

Figure 2: Sequence diagram of music streaming



Before any session can be created, at least one user having the mp3 file must sign up to distribute the song. First, the client will start a server and publish the relevant information to its user account using the function `edit_url(string url)`. The url string will contain the server's IP and port as well as the public key certificate used to encrypt all connections. With the server up and running the client signs up for distribution.

On the other side, the listener, having decided on a song, starts the process of creating a session. First, a distributor must be selected, the user might have a preference on which distributor to contact based on its previous experiences, but in case there is no preference the function `get_rand_distributor(bytes32 music)` is available. Then, a session is created using the function `create_session(bytes32 music, address distributor)` which maps the session id with the corresponding session structure. This identification value corresponds to the keccak256-hash value of the listener's and the distributor's address as well as the song's id which allows the listener to create multiple sessions with different distributors for the same music. This can become useful to maintain a song playing even if one distributor disconnects during the session. When the session is created the price of the song gets stored in the data structure in case it is modified while the session is running.

Also, the price amount as well as 10% more, the distributor fee, is transferred from the listener's balance to the session's balance. This assures the distributor that the entirety of the song could potentially be paid and, therefore, incentivizes him to maintain an efficient interaction which won't drive away the listener.

Finally, the transmission of the file is a repetitive process in which the listener pays for a chunk of the song and then request it to the distributor. The payment can simply be done using the function `get_chunk(bytes32 session, uint chunk_index)` which transfers, from the session balance, a fraction of the price to the author's balance and a fraction of the distributor fee to the distributor's. Then, the listener establishes a TLS connection to the distributor's server which is encrypted using its public key certificate. The listener sends the chunk request which includes the session id, the index and a signature using its chain address to prove its identity. The distributor then retrieves the session information from the smart contact checks that the index is paid and that the listener's address corresponds with the signature. If everything is correct, the distributor sends the requested chunk back. The listener finally checks that the chunk's keccak256-hash value corresponds to the one uploaded by the author to make sure the data received is authentic. This process is repeated for each chunk necessary to play the song.

When the user finishes or stops the song, the session is closed using the function `close_session(bytes32 session)`. Doing so will return any funds that were not used during the session to the user's account. If the listener never closes the session, the distributor is also allowed to do so because there is a possibility that the chunks will continue to be requested and appear as paid.

This whole process ensures that music can be distributed between users of the platform at the same time that the listener rewards both the author and the distributor. Each connection between clients is encrypted to ensure privacy as well as to defend users from tampering or man-in-the-middle attacks.

IV. Analysis

The proposed implementation, mentioned in the previous section, was tested on the local network of a single computer. The back-end consisted of a Hornet node and a Wasp node connected to each other. The front-end consisted of two python programs running in parallel, one taking the role of the distributor and the other the role of the listener. The objective of the following analysis was to measure the two most important metrics of this platform, the amount of gas and the execution time to perform basic actions. Both of these metrics are not controlled by users and can affect their experience as, if results are excessive, it will prevent the platform from offering proper music streaming.

The first metric to measure is the price paid by users to interact with the smart contract. The concept of gas is necessary because nodes protecting the ledger must be rewarded for the resources they provide. Unfortunately, it was impossible to estimate the gas costs of the implementation due to the development state of IOTA. The Web3.py library function to estimate gas costs, as of January 2023, is not compatible with the EVM implemented by IOTA as it does not yet have a way to retrieve historical gas information. MetaMask can estimate the costs without this information, but the results cannot be considered a realistic estimation as they are based on the Ethereum blockchain [20]. As IOTA continues to develop the Wasp node, ways to properly calculate gas-based fees are being tested on their networks [21]. Therefore, a realistic analysis of the gas costs of this paper's proposed solution cannot be made until a definitive solution is implemented.

The other important metric to measure is the time to execute any basic action. These values must be kept at a minimum to ensure a decent user experience. After analysing different execution times common to all actions, it is clear that functions that require the most attention are the ones that involve interaction with the smart contract. CPU execution times in the front-end are negligible as the python code used does not perform any complex computation. And the times measured for the transmission of data between users are also negligible as the testing environment is a local network. Interaction with the back-end can be divided into two groups: calls which are used to get the value of variables in the current smart contract state and transactions which are used to modify the smart contract state. Table I shows the times for each type of interaction from the moment the client sends the request to the moment there is a response. These results were obtained by sending the corresponding requests to the smart contract and measuring the average time to confirm.

Interaction	Result
Call	$\sim 0.1s$
Transaction	$\sim 0.9s$

Table I: Execution times of smart contract interaction

It is clear that L2 transactions are the processes that really increment the execution times. This might not be an issue for basic actions that only require one transaction as is the case for creating an account, transferring funds or uploading a song. But it can become problematic for managing a session due to the large amount of transactions required. To be precise, when a user listens to a full song, it is required to send one transaction to create the session, then as many transactions as the song has chunks, and finally a last one to close the session. The exact amount is therefore determined by the number of chunks that in turn is determined by how much data the client decides to include in each chunk when processing the mp3 file.

It is important to determine an optimal number of bytes per chunk as it will allow the platform to transmit songs faster as well as allow the user to only pay for the chunks used. To determine this value a test was conducted in which a 5-minute song, from a 2 MB mp3 file, was uploaded to the platform using multiple different amounts of bytes per chunk. In Table II, the different uploads were then fully transmitted and the

Data in KB	Transactions	Result
5	410	$\sim 10min$
15	138	$\sim 6min$
30	70	$\sim 4min$
60	36	$\sim 2min$

Table II: Music file transmission depending on data per chunk

number of transactions as well as the complete duration of execution was measured. The objective of this analysis was to find the smallest value allowing for the file to be transmitted faster than the song's duration. The results of this analysis determined that 30 KB per chunk is the value that will allow for the song to be divided into the most chunks while still allowing for the song to be transmitted faster than it can be listened to.

The analysis results show that the proposed solution can offer music streaming and at the same time manage the corresponding payments. However, this analysis can be subject to various criticisms. Mainly that the testing environment consisting of a local private network does not simulate the solution in a realistic environment. That is why, it is important to state that further analyses must be conducted that more closely simulate reality to reinforce the validity of the proposed solution. Finally, further analyses must also be conducted concerning the measurement of gas cost using future implementations of IOTA.

V. Conclusion and future work

The utilization of IOTA's distributed ledger technology presents an innovative approach for streaming music with synchronized payments. While there are implementation and privacy challenges that need to be addressed, the proposed solution holds great potential in benefiting musicians and audiences, empowering users to engage with music in a more secure and rewarding manner.

IOTA smart contracts effectively handle funds, resulting in a reduction of transaction volume and enhancing overall efficiency. IOTA ensures the security of the system by storing servers' public key certificates, enabling encrypted connections through TLS. Moreover, pseudonymity is preserved by employing wallet addresses, adding an additional layer of privacy to the platform. This proposed solution brings numerous benefits to musicians and audiences alike. Users have the ability to pay for requested parts and distribute music seamlessly, while musicians retain control over pricing and receive real-time rewards.

However, an implementation issue that needs to be addressed is the requirement for validation by a user with special permissions when dealing with music, potentially impacting efficiency and introducing a single point of failure in the system. Additionally, privacy concerns arise due to the storage of information in a public ledger. The potential risk of linking IP addresses with user behavior poses a challenge that needs to be carefully addressed to maintain user privacy.

References

[1] M. El-Hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, "A survey of internet of things (iot) authentication schemes," *Sensors*, vol. 19, no. 5, p. 1141, 2019.

[2] M. El-Hajj, M. Chamoun, A. Fadlallah, and A. Serhrouchni, "Analysis of authentication techniques in internet of things (iot)," in *2017 1st Cyber Security in Networking Conference (CSNet)*, pp. 1–3, IEEE, 2017.

[3] M. El-Haii, M. Chamoun, A. Fadlallah, and A. Serhrouchni, "Analysis of cryptographic algorithms on iot hardware platforms," in *2018 2nd Cyber Security in Networking Conference (CSNet)*, pp. 1–5, IEEE, 2018.

[4] M. Elhajj, H. Jradi, M. Chamoun, and A. Fadlallah, "Lasii: Lightweight authentication scheme using iota in iot platforms," in *2022 20th Mediterranean Communication and Computer Networking Conference (MedComNet)*, pp. 74–83, IEEE, 2022.

[5] Z. A.-A. Mohammad Fneish, M. El-Hajj, and K. Samrouth, "Survey on iot multi-factor authentication protocols: A systematic literature review," in *2023 11th International Symposium on Digital Forensics and Security (ISDFS)*, pp. 1–7, 2023.

[6] R. A. Rahimi and K.-H. Park, "A comparative study of internet architecture and applications of online music streaming services: The impact on the global music industry growth," in *2020 8th International Conference on Information and Communication Technology (ICoICT)*, pp. 1–6, 2020.

[7] M. Johnston, "How Spotify makes money: Premium Service generates the biggest share of revenue." https://www.investopedia.com/articles/investing/120314/spotify-makes-internet-music-make-money.asp, Mar 2022. [Accessed 29-Jan-2023].

[8] E. A. Drott, "Music as a technology of surveillance," *Journal of the Society for American Music*, vol. 12, no. 3, p. 233–267, 2018.

[9] D. Antal, A. Fletcher, and P. Ormosi, "Music streaming: Is it a level playing field?," *CPI Antitrust Chronicle*, 2 2021.

[10] D. Hesmondhalgh, "Is music streaming bad for musicians? problems of evidence and argument," *New Media & Society*, vol. 23, no. 12, pp. 3593–3615, 2021.

[11] S. Chavan, P. Warke, S. Ghuge, and R. V. Deolekar, "Music streaming application using blockchain," in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 1035–1040, 2019.

[12] J. Dimont, "Royalty inequity: Why music streaming services should switch to a per-subscriber model," *Hastings Law Journal*, vol. 69, pp. 675–700, 02 2018.

[13] C. Mullaney, A. Aijaz, N. Sealey, and B. Holden, "Peer-to-peer energy trading meets iota: Toward a scalable, low-cost, and efficient trading system," 2022.

[14] H. Farahani and H. R. Shahriari, "A privacy preserving iot data marketplace using iota smart contracts," 2022.

[15] R. Radhakrishnan and B. Krishnamachari, "Streaming data payment protocol (sdpp) for the internet of things," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1679–1684, 2018.

[16] G. S. Ramachandran, R. Radhakrishnan, and B. Krishnamachari, "Towards a decentralized data marketplace for smart cities," in *2018 IEEE International Smart Cities Conference (ISC2)*, pp. 1–8, 2018.

[17] M. El-Hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, "Ethereum for secure authentication of iot using pre-shared keys (psks)," in *2019 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 1–7, IEEE, 2019.

[18] L.-Y. Hou, T.-Y. Tang, and T.-Y. Liang, "Iota-bt: A p2p file-sharing system based on iota," *Electronics*, vol. 9, no. 10, 2020.

[19] K. R. Özyilmaz, M. Doğan, and A. Yurdakul, "Idmob: Iot data marketplace on blockchain," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pp. 11–19, 2018.

[20] "MetaMask User Guide: Gas." https://metamask.zendesk.com/hc/en-us/articles/4404600179227-User-Guide-Gas. [Accessed 29-Jan-2023].

[21] "IOTA Smart Contracts Release 0.3.0." https://blog.shimmer.network/iota-smart-contracts-release-030/, Sep 2022. [Accessed 29-Jan-2023].