THEMIS: Decentralized and Trustless Ad Platform with Reporting Integrity

Gonçalo Pestana  
Brave Software  
Panagiotis Papadopoulos  
Brave Software

Inigo Querejeta-Azurmendi  
Brave Software, University Carlos III Madrid  
Benjamin Livshits  
Brave Software, Imperial College London

Abstract

Online advertising fuels the (seemingly) free internet. However, even though users can access a website free of charge, there is a heavy cost on their privacy due to the deployed user tracking. To make matters worse, in the current ad ecosystem there is no transparency: apart from users, publishers and advertisers, there are numerous dubious middlemen exploiting the ad revenues and the users’ data beyond anyone’s control.

In this paper, we propose THEMIS: a novel privacy-by-design ad platform that is decentralized and requires zero trust from both users and advertisers. THEMIS (i) provides auditability to all participants, (ii) rewards users for interacting with ads, and (iii) allows advertisers to verify the performance and billing reports of their ad campaigns.

To assess the feasibility and practicability of our approach, we implemented a prototype using smart contracts and zero-knowledge schemes. Preliminary results of our evaluation show that during ad reward payouts, THEMIS can support more than 7M concurrent users on a single side-chain, a number that grows linearly by running multiple instances of THEMIS in parallel.

Keywords
Blockchain, Ad Platform, Trustless, Reporting Integrity, User Privacy, Zero Knowledge

1 Introduction

Digital advertising is the most popular way for funding websites, despite the many other alternative monetization systems that have been proposed [1–4]. Digital advertising revenues in the U.S. reached $57.9 billion during the first six months of 2019 [5], the highest revenue in history for the first half of any year, and a 17% year-over-year increase over the first half of 2018.

Web advertising though has fundamental flaws, including market fragmentation [6, 7], rampant fraud [8–14], and unprecedented privacy harm [15–18]. Further, more and more web users ignore annoying ads (click-through-rate: 2% on average [19]) that are trying to steal their attention [20] or even completely opt-out web advertising by using ad-blockers (47% of internet users globally, as of today) [21], thus costing publishers millions of dollars in ad revenues every year [22].

Academia and industry have responded by designing new monetization systems. These systems generally emphasize user choice, privacy protections, fraud prevention, or performance improvements. Brave Ads [23], Privad [24], and Adnostic [25] are three such recent systems. Despite the contributions of these systems, they have significant shortcomings that have limited their adoption. These systems (i) do not scale, (ii) require the user to trust central authorities within the system to process ad transactions, or (iii) do not allow advertisers to sufficiently measure campaign performance.

To make matters worse, as pointed out in [26], current advertising systems (iv) lack auditability: the ad network exclusively determines how much advertisers will be charged, as well as the revenue share the publishers will get. Malicious ad networks can overcharge advertisers or underpay publishers. What is more, (v) another issue is non-repudiation as bills cannot be adequately justified by the ad network and malicious advertisers can deny actual views/clicks and ask for refund.

1.1 Our Approach

To address these issues, in this paper, we propose THEMIS: the first decentralized ad platform that uses a side-chain design pattern and smart contracts to eliminate centralized ad network management. THEMIS eliminates the middlemen of the current ad ecosystem and at the same time requires zero trust from the participating entities allowing any of them to verify the proper execution of the protocol. THEMIS is designed to be practical for both users and advertisers. It protects user privacy while — following the paradigm of Brave Rewards [27] — it rewards users for viewing ads. Such user rewards policy aims to compensate users for the attention they give while interacting with ads. Existing, real world, reward schemes have triggered high user engagement, thus producing tremendous click-through-rates to their ad platforms (on average 14% compared to the 2% of the current ad ecosystem) [28].

THEMIS effectively addresses the auditability and non-repudiation issues pointed out above by providing a cryptographic tool-chest so participants can verify that everybody is following correctly the protocol. At the same time, THEMIS provides the advertisers with the necessary feedback regarding the performance of their ad campaigns but without compromising the end-user privacy. By guaranteeing the computational integrity of this reporting, advertisers can accurately learn how many users viewed their ads, but without learning exactly who.

1.2 Contributions

In summary, we make the following contributions:
(1) **System.** We propose THEMIS: a privacy-preserving advertising platform that compensates users for their attention and rewards them for ad viewing. Contrary to existing proposals, our system is decentralized and leverages smart contracts to orchestrate reward payments between users and advertisers. This way, our platform avoids relying on a trusted central authority.

(2) **Reporting integrity.** We leverage a partial homomorphic encryption scheme to (i) calculate ad payouts without revealing user interactions and (ii) provide every entity in the system a way to verify the validity of the payouts. To provide advertisers with the necessary feedback regarding the performance of their ad campaigns, we leverage a multiparty computation protocol to provide advertisers with reporting integrity while preserving the privacy of the users’ ad viewing.

(3) **Prototype implementation.** To assess the feasibility of our approach, we implement a prototype of our system in Rust and Solidity languages. We provide the source code of THEMIS publicly\(^1\).

(4) **Comparison of the different private payment protocols.** In the context of deploying THEMIS, we compare the performance of recently proposed zero-knowledge mechanisms that provide confidential and anonymous payments in Ethereum-based blockchains. We present comprehensive measurements of real-world deployments.

(5) **Performance evaluation.** We evaluate the performance of our system by conducting comprehensive scale-up experiments. As an optimization to the THEMIS protocol, we leverage the batching property of the on-chain confidential payment protocols tested above. Based on our preliminary measurements, we present how THEMIS can support around 7M concurrent users on a single side-chain, a number that grows linearly by running multiple instances of THEMIS in parallel.

2 Background

In this section, we provide the reader with the necessary knowledge regarding the techniques and mechanisms leveraged throughout the paper; we also describe why and how THEMIS uses them.

2.1 Proof of Authority Blockchains

THEMIS relies on a blockchain with smart contract functionality to provide a decentralized ad platform. Smart contracts enable us to perform all payments without trusting on a central authority. Ethereum Mainnet [29] is a such popular smart contract-based blockchain. However, due to its low transaction throughput, the high gas costs, and the overall poor scalability, in THEMIS, we chose to use a Proof-of-Authority (PoA) blockchain instead.

Consensus protocols constitute the basis of any distributed system. The decision on which consensus mechanism to use affects the properties, scalability and assumptions of the services build on top of the distributed system [30]. A PoA blockchain consists of a distributed ledger that relies on consensus achieved by a permissioned pool of validator nodes. PoA validators can rely on fast consensus protocols such as IBFT/IBFT2.0 [31, 32] and Clique [33], which result in faster minted blocks and thus PoA can reach higher transaction throughput than traditional PoW based blockchains. As opposed to traditional, permissionless blockchains (such as Bitcoin [34] and Ethereum [29]), the number of nodes participating in the consensus is relatively small and all nodes are authenticated. In our case the set of validators can consist of publishers.

**Private input transactions on PoA blockchains:** Providing private input functionality in smart contracts requires the inputs to be encrypted with all validator’s public keys. By encrypting both inputs and outputs with validator’s public keys, the parameters are private from readers of the public information, when at the same time validator nodes can run the smart contracts correctly and achieve consensus. To ease readability, we refer to the public keys of validators as a single one, \( \text{pk}_v \).

2.2 Cryptographic Tools

**Confidentiality:** THEMIS uses an additively homomorphic encryption scheme to calculate the ads payouts for each user, while keeping the user behavior (e.g. ad clicks) private. Given a public-private key-pair \( \text{pk}, \text{sk} \), the encryption scheme is defined by three functions: first, the encrypt function, where given a public key and a message, outputs a ciphertext, \( C = \text{Encpk} \cdot M \). Secondly, the decrypt function, that given a ciphertext and a private key, outputs a decrypted message, \( M = \text{Decsk} \cdot C \). And finally, the signing function, where given a message and a secret key, outputs a signature on the message, \( S = \text{Signsk} \cdot M \). The additive homomorphic property guarantees that the addition of two ciphertexts, \( C_1 = \text{Encpk} \cdot M_1, C_2 = \text{Encpk} \cdot M_2 \) encrypted under the same key, results in the addition of the encryption of its messages, more precisely,

\[
C_1 + C_2 = \text{Encpk} \cdot M_1 + M_2. \tag{1}
\]

Some examples of such encryption algorithms are ElGamal [35] or Paillier [36] encryption schemes.

**Integrity:** To prove correct decryption, THEMIS leverages zero knowledge proofs [37] which allow an entity (i.e., the prover) to prove to a different entity (i.e., the verifier) that a certain statement is true over a private input without disclosing any other information from that input other than whether statement is true or not. We denote proofs with the letter \( \Pi \), and use \( \Pi.\text{Verify} \) to denote verification of a proof.

**Distribution of trust:** THEMIS distributes trust to generate a public-private key-pair for each ad campaign, under which the sensitive information is encrypted. Thus, it uses a distributed key generation (DKG) protocol to share the knowledge of the secret. This allows a group of participants to distributively generate the key-pair \( \text{pk}_T, \text{sk}_T \) where each
participants has a share of the private key, $\text{sk}_i$, and no participant ever gains knowledge of the full private key, $\text{sk}_\text{f}$. The resulting key-pair is a $k-n$ threshold key-pair, which requires at least $k$ out of the $n$ participants that distributively generated the key, to interact during the decryption or signing protocols. We follow a similar distributed key generation (DKG) protocol as presented by Schindler et al. [38], which relies on smart contracts to decentralize public-key infrastructure [39]. To verify the correctness of the resulting decryption, we follow the protocol presented in [40].

In order to chose this selected group of key generation participants in a distributed way, THEMIS leverages Verifiable Random Functions (VRFs) [41, 42]. In general, VRFs enable users to generate a random number and prove its randomness. In THEMIS, we use VRFs to select a random pool of users and generate the distributed keys. Given a public–private key-pair $\text{VRF.pk}, \text{VRF.sk}$, VRFs are defined by one function: random number generation, which outputs a random number, $\text{VRF.Rand}$ and a zero knowledge proof of correct generation, $\Pi_{\text{VRF}}$.

$$\text{VRF.Rand}, \Pi_{\text{VRF}} = \text{VRF.RandGen}\text{VRF.sk}, \epsilon, \quad (2)$$

where $\epsilon$ is a random seed.

Confidential payments for account-based blockchains: Confidential payments on account-based blockchains allow transfer of assets between accounts without disclosing the amount of assets being transferred or the balance of the accounts. Additionally, the sender can prove the correctness of the payment (i.e., prove that there was no double spending). Confidential payments have drawn a lot of interest in both academia [43, 44] and industry [45, 46] recently. Throughout the paper, we consider the AZTEC [47] and the Zether [48] protocols as confidential payment protocols for THEMIS.

The AZTEC protocol implements a toolkit and a set of smart contracts for building confidential assets on top of the Ethereum Virtual Machine [19]. The AZTEC protocol defines a commitment scheme and zero-knowledge proofs for verifying and validating transactions without disclosing the balance of the asset transaction. An important feature of AZTEC is that it enables the prover to generate proof of correct payments in batches, which amortizes the costs of multiple payments.

The Zether protocol uses Sigma-Bulletproofs and one-out-of-many proofs to achieve confidential and anonymous payments in account-based blockchains. Zether does not provide batching of payment proof validations, which means that the time for settling payments grows linearly with the number of payments issued.

### 3.1 Main Actors

**PoA validator nodes:** THEMIS leverages a PoA side-chain that relies on consensus achieved by a pool of permissioned validator nodes. The validators’ role is of a miner of the side-chain, where they need to evaluate the smart contracts and mine new blocks.

**Campaign Facilitators (CF):** A CF interacts with advertisers to agree on an ad policy of their preference and deploys the smart contracts in the PoA side-chain. In Addition, the CF is responsible for performing the confidential and verifiable payments to the users.

**Advertisers:** The advertisers agree with the Campaign Facilitator the policies for each ad campaign they want to launch. They receive an anonymized feedback for the performance of their campaigns which integrity they can verify. Finally, they interact with the PoA chain to verify that the amount they are being charged for the users’ ad rewards indeed corresponds to their interactions.

**Users:** The users interact with the ads the advertising platform renders to them and generate interaction vectors. Next, they interact with the PoA chain so their rewards are computed and credited. THEMIS users may also participate in a consensus pool where they interact with other users in a peer-to-peer way. We refer to them as consensus participants (Con.Part).

### 3.2 Goals and Comparison with Alternatives

The goal of this paper is to design (i) a decentralized and (ii) trustless ad platform that (iii) is private-by-design while, at the same time, (iv) rewards users for the attention they give while viewing ads and (v) provides metrics for the ad campaigns of the advertisers. The key system properties we focus on while designing THEMIS, include privacy (for both user interactions and advertiser policies), decentralization and auditability (by providing verifiable rewards billing and campaign reporting integrity) and scalability:

1. **Privacy.** In the context of a sustainable ad ecosystem, we define privacy as the ability for not only users, but also advertisers to use our system without disclosing any critical information:
   
   (a) For the user, privacy means being able to interact (i.e., view, click) with ads without revealing their interests/preferences to advertisers or eavesdroppers. In THEMIS, we preserve the privacy of the user not only when they are interacting with ads but also when they claim the corresponding rewards for these ads.
   
   (b) For the advertisers, privacy means that they are able to setup ad campaigns without revealing any policies (i.e., what is the reward of each of their ad) to the prying eyes of their competitors. To achieve that, THEMIS keeps these ad policies confidential throughout the whole process, while it enables users to provably claim these rewards.
(2) **Decentralization and auditability.** Existing ad platforms [23–25] require a central authority to manage and orchestrate the proper execution of the protocol, either in terms of user privacy or billing. Although, as nicely pointed out in [26], what if this (considered as trusted) entity censors users by denying or transferring incorrect amount of rewards? What if it attempts to charge advertisers more than what they should pay based on users’ ad interactions? What if the advertising policies are not applied as agreed with the advertisers when setting up ad campaigns? One of the primary goals of our system is to be decentralized and trustless. To achieve this, THEMIS leverages a Proof-of-Authority (PoA) blockchain with smart contract functionality. To provide auditability, THEMIS leverages use zero-knowledge proofs to ensure the correctness and validity of both billing and reporting thus allowing all actors to verify the authenticity of the statements and the performed operations.

(3) **Scalability.** Ad platforms need to be able to scale seamlessly and serve millions of users. However, important proposed systems fail to achieve this [24, 25]. In this paper, we consider scalability an important aspect affecting the practicability of a system. THEMIS needs to not only serve ads in a privacy preserving way to millions of users but also finalize the payments related to their ad rewards.

### 3.3 Threat Model

In THEMIS, we assume computationally bounded adversaries capable of snooping communications, performing replaying attacks, forging signatures or cheating by not following the protocol in an honest way.

One such adversary may act as CF aiming to collect more processing fees than agreed at the cost of user rewards or advertiser refunds. We assume that such adversary can control any node of the system but (following the limitations of the underlying PoA) not the majority of the validator nodes.

Other adversary may aim to breach privacy of the user and snoop the interactions of the user with the ads. This user information could reveal interests, political/religious preferences, that can later be sold or used beyond the control of the user [50–53]. We assume that such adversary can control any node in the system but cannot control all users. We assume that it controls at most $k$ of the $n$ randomly selected users that are part of the consensus pool (due to the threshold key they hold, see Section 2.2).

There may be also adversaries that will try to break the confidentiality of the advertisers’ agreed ad policies. Revealing such information could disclose rewarding strategies and give an advantage to competitors. We assume that such an adversary may act as a user and/or advertiser in the protocol but cannot control the campaign facilitators or the PoA validators. Table 1 summarizes the assumptions required for each goal of our system.

| Entities       | User privacy | Ad-policy privacy | Auditability |
|----------------|--------------|-------------------|--------------|
| CF             | No           | Yes               | No           |
| PoA Validators | No           | Yes               | Majority     |
| Com.Part       | $k$-out-of-$n$ | Yes               | No           |

Table 1: Trust assumptions of the key actors of THEMIS protocol related to each of the properties our scheme offers.

**Out-of-scope Attacks:** We acknowledge that client-side fraud is a real problem in current ad ecosystem [10, 11, 54] and the majority (if not all) the alternatives [24, 25, 55]. In this paper, we do not attempt to address this problem. There is a significant body of research already aiming to detect and deal with such attacks (i.e., bot clicks, click farms, automated agents) [56–59] and therefore we consider these attacks beyond the scope of this work. To mitigate their impact and similar to other rewarding ad platforms [27], in THEMIS, we introduce a cap on the reward requests per day per user.

### 4 System Overview

In this section, we present in detail the decentralized and privacy-preserving advertising system. We begin with a strawman approach to describe the basic principles of the system. We build on this straw-man approach and, step-by-step, we introduce the decentralized ad platform, THEMIS. For presentation purposes, in the rest of this paper, we assume that users interact with THEMIS through a web browser, although users may interact with it through a mobile app in the exact same way.

#### 4.1 A Straw-man Approach

Our straw-man approach is the first step towards a privacy-preserving and trustless online advertising system. Our goal at this stage is to provide a mechanism for advertisers to create ad campaigns and to be correctly charged when their respective ads are delivered to users. In addition, the system aims at keeping track of the ads viewed by users, so that (i) advertisers can have feedback about their ad campaigns and (ii) users can be rewarded for interacting with ads. All these goals should be achieved while preserving ad policy privacy and user privacy.

We assume three different roles in this system: (i) the users, (ii) the advertisers, and (iii) an Ad Campaigns Manager (CM). The users are incentivized to view and interact with ads created by the advertisers. The CM is responsible (a) for orchestrating the protocol, (b) for handling the ad views reporting and finally (c) for calculating the rewards that need to be paid to users according to the policies defined by the advertisers. Table 2 summarizes the notation used throughout this section.

#### 4.1.1 Privacy-preserving Ad Personalization

To perform privacy-preserving ad personalization, in our system we follow the paradigm of Adnostic [25] and the ad platform of Brave Ads [23] which has been in continuous operation since 2019 [55]. This way, in our system, the user downloads (e.g., periodically or while on WiFi) from the ad
server (maintained by the CM) the most recent version of a vector, namely ad catalog, that includes ads from all active ad-campaigns and the name of the advertiser who created it.

The ad-matching happens locally based on a pre-trained model and the user’s interests extracted from their web browsing history in a similar way as in [23, 24]. Zero data leaves the user’s device, thus creating a walled garden of browsing data that are used for recommending the best-matching ad while user privacy is guaranteed.

The idea of prefetching ads is not new [24, 25, 55, 60, 61]. Studies have shown that prefetching ads in bulk for smartphone users is not only practical but can reduce energy consumed during ad transactions by more than 50% [62].

### 4.1.2 Incentives for Ad-viewing
Following the paradigm of existing academic and corporate user rewarding schemes [23, 63, 64], in THEMIS, we compensate users for the attention they pay to ad impressions, thus incentivizing them to interact with ads. Already launched ad rewarding schemes increased the user engagement, providing this way high click-through-rates (i.e., 14% on average [28]) to the served ads.

In THEMIS, each viewed/clicked ad yields a reward (this can be fiat money, crypto-coins, coupons, etc.). Different ads may provide different amount of reward to the users. This amount is agreed by the corresponding ad creator (i.e., the advertiser) and the CM. The reward claiming takes place periodically (e.g., every 2 days, every week or every month) when the user requests their reward for the ads they interacted with. In Figure 1, we present an overview of the reward defining and claiming procedure of this straw-man approach.

**Phase 1: Defining Ad Rewards:** In order for an advertiser to have their ad campaign included in the next version of the ad catalog, they first need to agree with the CM on the policies of the given campaign (i.e., rewards per ad, ad impressions per user, etc.) (step 1 in Figure 1). Then, the CM encodes the agreed policy as a vector \( P \), where each index corresponds to the amount of tokens that an ad yields when viewed/clicked (e.g., Ad 1: 0.4 coins, Ad 2: 2 coins, Ad 3: 1.2 coins). The CM stores this vector privately and the advertiser needs to trust that the policies are respected. The indices used in \( P \) are aligned with the ones of the ad catalog.

For the sake of simplicity, throughout this section, we consider one advertiser who participates in our ad platform and runs multiple ad campaigns. Of course, in a real world scenario many advertisers can participate running many ad...
campaigns simultaneously. We also consider as “agreed policies” the amount of coins an ad provides as reward to a clicking user.

**Phase 2: Claiming Ad Rewards:** The user on their end, create locally an interaction vector where information is noted about the number of times each ad of the catalog was viewed/clicked (e.g., Ad 1: was viewed 3 times, Ad 2: was viewed 0 times, Ad 3: was viewed 2 times). In every payout period, the user encrypts the state of the interaction vector. More technically, let \( ac \) be the interaction vector containing the number of views/clicks of users with each ad, where element \( i \) of vector \( ac \) represents the number of times ad \( i \) was viewed/clicked. On every payout period, the user generates a new ephemeral key pair \( pk, sk \), to ensure the unlinkability of the payout requests. By using this key, they encrypt each entry of \( ac \):

\[
EncVec = [Encrypt(pk_1, \ldots, Encrypt(pk, n_{n,A})]
\]

and send \( EncVec \) to the CM (step 2a in Figure 1).

CM cannot decrypt the received vector and thus cannot learn the user’s ad interactions (and consequently their interests). Instead, they leverage the additive homomorphic property of the underlying encryption scheme (as described in Section 2.2) to calculate the sum of all payouts based on the interactions encoded in \( EncVec \) (step 2b in Figure 1). More formally, the CM computes the aggregate payout for the user as follows:

\[
Aggr.Res = \sum_{i=1}^{N} P_i[i] \cdot EncVec[i],
\]

where \( P_i \) is the ad policy associated with the ad in the position \( i \) of the vector. Then CM signs the computed aggregate result:

\[
SignReward = Sig.SignAggr.Res, sk_{CM}
\]

and sends the 2-tuple \( Aggr.Res, SignReward \) back to the user.

Upon receiving this tuple (step 2c in Figure 1), the user verifies the signature of the result: \( Sig.VerifyAggr.Res, SignReward \). If the signature is not valid, the user can prove to other participants of the consensus pool to be decrypted. Each consensus participant performs the decryption result:

\[
Dec.Aggr.Res = Decryptsk, Aggr.Res
\]

As a final step, it proves the correctness of the performed decryption by creating a zero knowledge proof of correct decryption: \( \Pi_{Res} \).

**Phase 3: Payment Request:** Finally, the user generates the payment request and sends the following 4-tuple to the CM (step 3a in Figure 1):

\[
(Dec.Aggr.Res, Aggr.Res, SignReward, \Pi_{Res})
\]

As a next step (step 3b in Figure 1), the CM verifies that the payment request is valid. More technically, CM will reject the payment request of the user if

\[
Sig.Verifypk_{CM}, SignReward, Aggr.Res = \bot
\]

or

\[
Verify\Pi_{Res} = \bot
\]

Otherwise, it proceeds with transferring the proper amount (equal to \( Dec.Aggr.Res \)) of reward to the user.

**In summary:** Till now, we have presented a system which guarantees that:

1. The user received the rewards they earned by interacting with ads. This happened without requiring the user to disclose which ads they clicked on.
2. CM is able to correctly apply the pricing policy of each ad without disclosing any information to users or potential competitors of the advertiser.

### 4.1.3 Reporting to Advertisers

Apart from providing the users with the proper incentives to interact with ads, in order to make an ad-platform practical, feedback about the ad campaigns must be provided to the advertisers, too. What is more, during billing procedure they need to be able to verify the integrity of the reported statistics by the CM regarding the number of times an ad was viewed/clicked by the users. Based on these statistics, advertisers get rewarded depending on the times their corresponding ad was clicked throughout the campaign. To achieve this, whenever a new version of the ad-catalog is online and retrieved from the users, a new key-pair, \( pk_T, sk_T \), is generated. This key is used to encrypt a copy of the \( ac \) vector CM (Step 2a in Figure 1). Hence, apart from the EncVec illustrated in Equation 3, each user also sends EncVec’ to the CM, where:

\[
EncVec' = [Encrypt(pk_T, n_1, \ldots, Encrypt(pk_T, n_{n,A})]
\]

The key used here, \( pk_T \), is a public threshold key generated in a distributed way, common to all users. In order to generate such a key, a pool of multiple participating users\(^2\), namely consensus pool, is created (more details on how the consensus pool is chosen in Section 4.2.3). For this purpose, the consensus pool runs a Distributed Key Generation (DKG) algorithm [65]. This results in a shared public key \( pk_T \) and each consensus pool participant owning a privacy key share \( sk_{T,i} \). The public key, \( pk_T \), is sent to the CM to make it accessible to all users.

Whenever CM needs to measure the effectiveness of the current version of the ad-catalog, it merges the reported EncVec’ of every user into a single report. This report includes the total number of interactions that each ad received by all users. During this merge, the CM performs homomorphic addition of all the reported EncVec’, leveraging the fact that every user used the same key for the encryption:

\[
Aggr.Cliks = \sum_{i \in U} EncVec'_i
\]

where \( U \) is the set of users and \( EncVec'_i \) is the vector corresponding to user \( i \). Then, the CM sends \( Aggr.Cliks \) to the participants of the consensus pool to be decrypted. Each consensus participant \( i \) responds back with a tuple that includes (a) its partial decryption:

\[
Dec.Share_i = Part.Decsk_{T,i}, Aggr.Cliks
\]

and (b) a proof of correct decryption. As soon as CM collects at least the threshold \( k \) of such tuples, it verifies the received proofs and combines all decryption shares together to fully decrypt the ciphertext. The decrypted vector reveals the number of times each ad was clicked by all users. This

\(^2\)Users are incentivized to participate in this pool. Details on how to orchestrate the incentives are left outside the scope of this paper.
information is publicly available and is used by advertisers to validate how much they were charged per ad campaign.

4.2 THEMIS: A Decentralized Ad Platform

In Section 4.1 above, we presented the core functionality of our ad-platform. We described (i) how the user gets rewarded for their ad interactions, and (ii) how the advertiser can verify the integrity of the billing and performance report of their ad campaigns.

However, the centralization and the lack of auditability of this straw-man approach creates significant limitations with respect to the goals and threat model described in Section 3:

- Advertisers need to blindly trust the CM with the full custody of the rewards budget set for each ad campaign.
- Users and advertisers have to trust that the CM respects the agreed policies during payouts and transfers the correct amount of rewards (step 3c in Figure 1).

This means that (similarly to existing approaches [24, 25]) the entire protocol relies on how trustworthy the central authority (the CM in our straw-man approach) is.

4.2.1 The THEMIS protocol

To address these issues in THEMIS, we leverage a distributed PoA ledger, where business and payment logic are orchestrated by smart contracts. All participants of THEMIS can verify that everyone runs the protocol correctly, thus requiring zero trust from each other. In particular, we define two smart contracts (See Appendix A for full detail and code):

1. The Policy Smart Contract (PSC), which is responsible for the billing of users’ rewards and validating the payment requests. Furthermore, it is in this smart contract that $Enc P$ is stored.
2. The Fund Smart Contract (FSC), which receives and escrows the funds needed to run the campaign. The FSC is responsible for releasing (i) the funds needed for the user rewards, (ii) the advertiser refunds, and (iii) the processing fees for the CF.

In THEMIS, instead of the central trusted authority of the CM, we introduce the role of a Campaigns Facilitator (CF). The responsibilities of the CF are to (i) negotiate the policies (e.g., rewards per ad, impressions per ad) of the advertisers; (ii) deploy smart contracts in the PoA ledger; and, lastly, (iii) handle the on-chain payments. In the THEMIS model, the CF is an authorized entity chosen by the blockchain’s PoA validators (or part of the consortium), which could be an independent third party like the EFF [66] or a trustworthy industry body. Our system ensures that everybody can audit and verify the behaviour of the different CFs, so advertisers can pick the CF they prefer to collaborate with based on their reputation. The CF is incentivized to perform the tasks required to facilitate the ad catalog, by receiving processing fees from advertisers.

Phase 1: Defining ad rewards: In Figure 2, we present a high-level overview of the reward claiming procedure of THEMIS. Similar to the straw-man approach presented in Section 4.1.2, in order for an advertiser to include their ad campaign in the next ad-catalog facilitated by the CF of their preference, they need to transmit their policies (e.g., reward per ad) to the CF (step 1a in Figure 2). In order to achieve that, each advertiser exchanges a symmetric key\(^3\) for each ad campaign $S_i$ with the CF. It then encrypts the corresponding ad campaign and sends it together with their ad creatives to the CF. On their end, the CF (i) decrypt and check the policies are as agreed; (ii) merge the encrypted policies of the different advertisers into the encrypted policy vector, $Enc P$; and (iii) deploy the two public smart contracts for this ad-catalog version (step 1b in Figure 2). In addition, the CF (iv) creates a vector $S$ with all the advertisers’ secret keys $S_i$:

$$S = [S_1, S_2, \ldots, S_{n_A}]$$ (11)

and (v) generates a vector, $Enc S$, that includes each of the elements in $S$, encrypted with the public key ($pk_V$) of the PoA validator nodes:

$$Enc S = [Enc_{pk_V}, S_1, \ldots, Enc_{pk_V}, S_{n_A}]$$ (12)

Then, the CF (vi) stores $Enc S$ in PSC to allow the PoA validators to decrypt and apply the corresponding policies on users ad interaction vectors.

Once PSC is deployed, the advertisers must verify if $Enc S$ encodes the policies agreed with CF (step 1c in Figure 2). More specifically:

1. First, the advertisers fetches $Enc P$ vector from the public storage of the PSC and decrypts the policy, $Enc P_i$, using the corresponding symmetric key, $S_i$, and verifies it is the agreed value.

2. Second, they fetch the escrow account address from FSC and transfer funds to the escrow account. The amount of funds needed is determined by the number of impressions they want per ad, its part of the agreed policy $p_i$, and the processing fees to pay CF. Once the campaign is over, the advertisers may get a refund based to the final number of impressions viewed/clicked by users. By staking the campaign’s funds, the advertiser is implicitly validating the deployed ad policies.

Once the FSC verifies that advertisers have transferred to the escrow account the correct amount of funds, the campaign is initialized and verified.

Phase 2: Claiming ad rewards: Similar to what illustrated in Section 4.1, in order to claim their ad rewards, each user creates an ephemeral key pair $pk, sk$ and obtains the public threshold key $pk_T$ generated by the consensus pool (in Section 4.2.3, we describe in detail how the consensus pool is generated). By using these two keys, each user encrypts their ad interaction vector to generate two ciphertexts: (a) the $EncVec$ that is used to claim ad rewards and (b) the $EncVec'$ that is used for the advertisers reporting.

Contrary to our centralized straw-man approach, in THEMIS, the aggregate calculation is performed via a PSC (as can be seen in step 2b in Figure 2). Thus, the user calls a public endpoint on PSC and transmits both ciphertexts. To

\(^3\)For the creation of this key, they follow the Diffie-Hellman key exchange protocol [67]
Campaigns Facilitator
Advertiser picks a
Advertiser
(1c) Advertiser verifies smart contract and funds escrow account
(1b) Campaigns Facilitator deploys (1) PSC and (2) FSC
(1a) Advertiser forwards ad-campaign’s policies
(2a) User sends their encrypted ad interactions
(2b) PSC calculates encrypted aggregate
(3) User requests payment
(4a) FSC releases rewards, fees and refunds
(4b) Campaigns Facilitator transfers the rewards to the users based on their ad interactions
(4c) Campaigns Facilitator returns unused funds

Figure 2: High-level overview of the user rewards claiming procedure in THEMIS. This operation of THEMIS consists of 4 different phases: (1) Definition of Ad Rewards and set policies, (2) Ad Reward claiming, (3) Payment Request, (4) Settlement of user payments and advertiser refunds.

calculate the encrypted sum of the rewards the user can claim (step 2b in Figure 2), a PoA validator runs PSC as follows: (i) it decrypts each policy \( P_i \) using \( \text{Enc} S \) (here, THEMIS leverages the private input transactions 2.1); (ii) applies on \( \text{EncVec} \) ciphertext the additively-homomorphic property of the underlying encryption scheme (as shown in Equation 4); and (iii) stores the result (i.e., \( \text{Aggr.Res} \) ) in the smart contract public store.\( ^4 \)

Phase 3: Payment request: Once the PSC calculated the aggregate result (step 3 in Figure 2), the user generates a payment request, \( \mathcal{E} \), that, if valid, is published in FSC. More technically, the user (i) creates an ephemeral blockchain account (used only once per request) with address \( \text{Addr} \) and then (ii) fetches and decrypts \( \text{Aggr.Res} \) to get the decrypted reward, \( \text{Dec.Aggr.Res} \), and generates the proof of correct decryption, \( \Pi_{\text{Res}} \). This way, the user (iii) generates the payment request which consists of the following 3-tuple:

\[
\mathcal{E} = [\text{Dec.Aggr.Res}, \Pi_{\text{Res}}, \text{Addr}] .
\]  

(13)

Then, (iv) the user calls a public endpoint on PSC with the \( \mathcal{E} \) encrypted with the validators keys, \( \text{Encpk}_{V_i} \mathcal{E} \) as input. The function then fetches the user’s aggregate, \( \text{Aggr.Res} \), decrypts the request, and verifies the zero knowledge proof, \( \Pi_{\text{Res}} \). If the proof is valid, it stores \( \text{Addr} \) in the FSC, which keeps a list of buffered user payments until marked as paid.

Phase 4: Payment settlement: The final step of the protocol regards the settlement of the user payment and advertiser refund.

Specifically, the settlement of the user rewards in THEMIS needs to happen in a confidential way to preserve the privacy of the total of earned reward. To achieve this, CF fetches the pending payments requests from FSC, and calculates the total amount of funds required to settle all pending payments.

As a next step (step 4a in Figure 2), the CF calls a public function of FSC requesting to transfer (to an operational account owned by CF) a given amount of tokens needed to cover the payments. If the CF misbehaves (by requesting an incorrect amount of tokens), it will be detected, and either advertisers or users will be able to prove its misbehaviour.

Finally, CF settles each of the pending reward payments by using a confidential payment scheme 2.2. After finalizing the payments correctly (and if there are no complaints form either users or advertisers), the CF receives from FSC the processing fees.

In case of unused staked funds, the advertisers need to be refunded. To achieve this (step 4c in Figure 2), FSC utilizes the aggregate clicks per ad vector that the consensus pool has computed during the advertisers reporting (see Section 4.1.3). Based on this vector and the agreed rewards, the FSC proceeds with returning to the advertisers the unused funds.

4.2.2 Misbehaving Campaigns Facilitator

The CF can cheat in two ways: (1) as it is the entity orchestrating the confidential payments, it may send incorrect rewards to users or (2) it could use its power to send rewards not only to the user but to other accounts of their control. Both of these actions may be discovered by either users or advertisers.

(1) In case of scenario (1) the users can provably challenge CF for incorrect behaviour by proving that the payment received does not correspond to the payment request they generated. To do so, the user calls the FSC to prove that the amounts received by the private payment does not correspond to the decrypted aggregate in the payment request \( \mathcal{E} \). We stress that in case a user must undergo such a scenario, only the aggregate amount of a single ad-catalog will be disclosed (and not its interaction with ads).

(2) In case of scenario (2), the escrow account will not have enough funds, resulting in some advertiser getting a smaller refund to what is stated in the performance report of their ad campaign (as described in Section 4.1.3). In this case, the advertiser can prove that the received refund does not correspond to the amount staked in Phase 1 (see Section 4.2.1) minus the rewards paid to users based on the numbers of clicks their ads received.\( ^{\text{4}}\)
To claim misbehaviour, users and advertisers file the complaint via a public function on FSC (that validates the complaint). If any of the complaints happen, the FSC switches its state as “failed” and CF will not receive any processing fees, something that affects their reputation.

### 4.2.3 Consensus Pool Selection

Throughout our system description, we rely on a consensus pool to distributedly generate a threshold key-pair that is used for creating the performance reports of the advertisers’ ad-campaigns. This consensus pool consists of a number of selected users that have opted-in and a smart contract used to orchestrate the process of defining this pool (as proposed in [38]). Any user can opt-in in the draw to become a consensus pool participant, and a random subset of all participating users is selected. Specifically, the smart contract keeps a time interval during which users who want to opt-in can register as participants for the draft. After that, the smart contract utilizes an external oracle to select a random seed, $\epsilon$, which is used to generate random numbers.

Every registered user generates an ephemeral Verifiable Random Functions key-pair, $\text{VRF.pk}, \text{VRF.sk}$, and publishes the public key in the smart contract’s public store. Once the registration phase is closed, the smart contract calculates a threshold, $\text{MAX.DRAW}$

$$\text{MAX.DRAW} = \frac{n}{L} \cdot p$$  \hspace{1cm} (14)

where $L$ is the size of the drawing pool (formed by all opted-in users), $n$ being the expected number of participants in the distributed key generation, and $p$ being an integer such that $\mathbb{Z}_p$ is the space of random numbers outputted by $\text{VRF.RandGen}$. Note the use of the $\epsilon$.

Next, the participating users calculate their corresponding random number and a proof of correct generation

$$\text{VRF.Rand}, \Pi^{\text{VRF}} = \text{VRF.RandGen}\text{VRF.sk}, \epsilon.$$  

All participants with $\text{VRF.Rand} < \text{MAX.DRAW}$ have passed the selection step, and proceed to publish $\text{VRF.Rand}, \Pi^{\text{VRF}}$ in the smart contract. They are then required to proceed with the distributed key generation step defined in 4.1.3. To avoid cases where there are not enough participants available online, in THEMIS the participation in the consensus pool is incentivised.

### 4.3 Implementation Details

We implemented the core components necessary to run the main THEMIS stages described in the Section 4.2.1. More specifically, the implemented software supports the Phase 1 (defining ad rewards on the client side), Phase 2 (claiming ad rewards) and Phase 3 (issuing and verifying the payment request) of the protocol. We measure the performance and scalability of THEMIS using this implementation. The source code of the client and smart contracts has been made publicly available in Github [68].

**Smart contracts implementation:** We implemented the policy smart contract in Solidity [69]. The cryptographic computations required to calculate the rewards over encrypted data and verify the proof of correct decryption leverage the pre-compiled smart contracts [70]. These pre-compiled smart contracts implement the addition and scalar multiplication over the $\text{alt_bn128}$ curve, improving the performance of our homomorphic operations.

**Client-side implementation:** We implemented the client logic in Rust. The client-side implementation leverages the $\text{web3-rust}$ [71] crate to interact with the smart contracts. In addition, we used both $\text{curve25519-dalek}$ [72] and $\text{elgamal_ristretto}$ [73] crates to implement the underlying public-key cryptography, and corresponding operations, required by THEMIS.

### 5 Performance Evaluation

In this section, we set out to explore the performance and scalability of THEMIS. More specifically, we measure the execution time of the different components of our system that include heavy cryptographic operations running in smart contracts deployed on a single side-chain. In addition, we measure the execution time of the distributed key generation and decryption of the consensus pool. Building on these measurements, we evaluate the overall scalability of THEMIS and assess how the scalability improves linearly with the number of parallel deployed side-chains.

In order to measure the performance of the smart contracts responsible for the on-chain computations and the confidential payments below, we used the Mjölir tool [74] to deploy a Quorum side-chain in a production-like environment. We deployed a 4x Quorum side-chain on AWS, each node running on an EC2 t.2xlarge instances, all in the same region and part of the same subnet. For the purpose of the measurements, the network communication is considered negligible. This setup can be easily emulated in production by setting up peering among different AWS Virtual Private Clouds across validator organizations. The protocol configured for the validator nodes to reach consensus was the Istanbul Byzantine Fault Tolerant (IBFT) consensus protocol [31].

**Client side performance:** To generate a payment request, the client needs to (i) encrypt the interaction array (for our evaluation, the length of this array is 128 entries), step 2a in Figure 2, and (ii) decrypt the aggregate and recover the plaintext (the aggregate is a single decryption), step 3 in Figure 2. We see from Figure 3 that the overhead to generate reward requests can be done on a commodity laptop or mobile device, without impacting user experience and energy consumption. Similarly, fetching, decrypting and recovering the reward also produces a low overhead, as can be seen in Figure 4. Moreover, the payment requests are performed on the client-side at relatively long intervals (e.g. daily, weekly or even monthly). Hence, we conclude that the overhead on the client is negligible.

---

5 We deploy this smart contract in the public Ethereum Main-net; the number of interactions with the blockchain is small so the gas cost and latency are minimal.

6 The process of accounting and incentivising the participants of the consensus pool is out of scope of this paper.
Aggregate Ad Clicks Computation: In THEMIS, a consensus pool is responsible for computing the aggregated clicks each ad received by the users in order to assess the performance of each ad-campaign. In the next experiment, we measure the overhead THEMIS imposes to the users who participate in the consensus pool and Figure 5 presents the results for the different threshold levels and number of participants. The blue line shows the execution time of distributively generating the public key and the key shares. For a majority of 80-of-100, the distributed key generation process takes 11.4 seconds. This considers the upper bound as we assumed that participants perform their tasks one after the other. In practice, these tasks are executed in parallel. Note, that this computation has to be performed only once for each version of the ad catalog. The operation of the provable decryption of the overall aggregate clicks vector (red line) takes around 1.2 seconds. In addition, decrypting the overall aggregate vector is required only once per ad catalog.

Comparison of the different confidential transaction protocols: For the settlement of the rewards payments, in THEMIS we need a confidential transaction protocol to ensure the confidentiality and integrity of the performed user payments. We set out to compare the performance of the two most popular such protocols in the context of THEMIS: AZTEC [47] and Zether [48]. Our goal is to understand how many confidential transactions can be issued per unit of time on a PoA blockchain and assess whether confidential payments can constitute a bottleneck to the scalability of THEMIS.

While in Zether the proof validation time increases linearly with the amount of payments, AZTEC implements a proof batching mechanism, which enables senders to amortize the time of proof verification in the smart contract by performing multiple payments simultaneously. In Figure 6, we measure the throughput of AZTEC for the different batch sizes. As we can see, AZTEC in the case of THEMIS can achieve around 239K payments/day for a batch size of 500 proof verifications. Contrary to that, we measured the throughput that Zether achieves and we see that it reaches 10 payments per minute (or 600 payments per hour, 14,000 payments per day). This means that for the case of THEMIS, the throughput of 14,000 confidential payments/day falls short from our scalability requirements.

In Figure 7, we compare the time it takes for AZTEC and Zether to finalize payments. As we can see, in case of Zether the required time increases linearly with the growing number of payments. On the other hand, in case of AZTEC, this is constant: 3 seconds, due to its batching feature. Our measurements confirm that we can achieve scalable confidential payments in the context of THEMIS by leveraging AZTEC.
Figure 6: Confidential payments per day for the different batch sizes while using the AZTEC protocol. The throughput reaches more than 239K payments/day for a batch size of 500 proofs.

Figure 7: Required time for AZTEC and Zether to finalize payments. In Zether the processing time increases linearly with the growing number of payments. On the other hand, in AZTEC, this is constant, 3 seconds.

5.1 System Scalability

We measured the performance and scalability of the reward calculation phase of THEMIS. More specifically, we measure the time it takes for the client and smart contract to participate in the THEMIS protocol, and to calculate the ad reward and verify the proofs of correct behaviour respectively.

To that end, we have setup a testing environment that starts 50 concurrent clients. The clients interact with the smart contracts at roughly the same time. The clients request a reward calculation over a catalog of 128 ads. Using this setup, we measure the time it takes for individual clients to complete the reward calculation, proof verification and end-to-end performance of the protocol.

Figure 8 shows the accumulated time in seconds for different phases of the reward calculation by the client and smart contract, with an ad catalog of size 128. The results show that:

1. It takes about 17 seconds for the client to request the reward calculation and retrieve the encrypted aggregate from the smart contract.
2. It takes about 18 seconds for performing (i), and for the client to decrypt the aggregate and perform the plaintext recovery locally;
3. Finally, it takes about 20 seconds for the client and the smartEVM to perform (ii) and for the client to submit the decrypted aggregate and proof of correct decryption and, finally, for the smart contract to verify the correctness of the user input.

We conclude that when 50 users concurrently request a reward from the slide-chain, it takes around 20 seconds to complete the protocol per client, using an ad catalog of 128 ads. In terms of overall scalability, we conclude that THEMIS can process about 7 Million users reward requests per month.

5.2 Horizontal Scaling and Multiple Side-chains

We see although the computations performed in smart contracts of THEMIS are highly parallelizable, the one-threaded event loop of the Ethereum Virtual Machine (EVM)\(^7\) cannot support parallel and concurrent computations, thus becoming a scalability bottleneck.

To overcome this shortcoming we envision THEMIS operating on top of multiple parallel side-chains that will be responsible for one or more ad catalogs each. Although this could increase coordination complexity, the scalability gains are considerable as the number of ads processed grows linearly with the number of side-chains supporting this way simultaneously billions of users.

\(^7\)The EVM is the run-time virtual machine where the smart contract instructions are executed in each of the validator’s machine.
6 Related Work

The current advertising ecosystem abounds with issues associated with its performance, its transparency, the user’s privacy and the integrity of billing and reporting. These failures are already well studied and there are numerous works aiming to shed light on how digital advertising works [20, 75–80].

Apart from the studies highlighting the failures of current ad delivery protocols there are also important novel ad systems proposed. In [81], Juels is the first to study private targeted advertising. Author proposes a privacy-preserving targeted ad delivery scheme based on PIR and Mixnets. In this scheme, advertisers choose a negotiant function that assigns the most fitting ads in their database for each type of profile. The proposed scheme relies on heavy cryptographic operations and therefore it suffers from intensive computation cost. Their approach focuses on the private distribution of ads and does not take into account other aspects such as view/click reporting.

In [25], authors propose Adnostic: an architecture to enable users to retrieve ads on the fly. Adnostic prefetches \( n \) ads before the user starts browsing and stores them locally. Aside from the performance benefits of this strategy, Adnostic does this prefetching also in order to preserve the privacy of the user. The parameter \( n \) is configurable: larger \( n \) means better ad matching, while smaller \( n \) means less overhead. In order for the ad-network to correctly charge the corresponding advertisers, Adnostic performs secure billing by using homomorphic encryption and zero-knowledge proofs.

In [24, 82, 83], authors propose Privad: an online ad system that aims to be faster and more private than today’s ad schema. Privad introduces an additional entity called Dealer. The Dealer is responsible for anonymizing the client so as to prevent the ad-network from identifying the client and also handle the billing. To prevent the Dealer from accessing user’s behavioral profile and activity it encrypts the communications between the client and the Dealer. A limitation of Privad is that Dealer is a centralized entity that needs to be always online.

In [84], authors propose OhliviAd: a provably secure and practical online behavioral advertising architecture that relies on a secure remote co-processor (SC) and Oblivious RAM (ORAM) to provide the so called secure hardware-based PIR. In OhliviAd, to fetch an ad, a user first sends their encrypted behavioral profile to the SC which securely selects the ads that match best based on the algorithm specified by the ad network. To prevent the ad-network from learning which ads are selected, they leverage an ORAM scheme. The selected ads are finally sent to the user encrypted, along with fresh tokens used to billing. User will send back one of these tokens as soon as they view/click on an ad.

In [26] authors point out that, in current advertising systems the ad-network exclusively determines the payment to get from advertisers and the revenue to share with publishers. This means that (i) a malicious ad-network can overcharge advertisers or underpay publishers. To make matters worse, as bills cannot be justified by the ad-network, malicious advertisers can deny actual views/clicks to ask for refund. On the other hand, (ii) malicious publishers may claim clicks that did not happen, in order to demand higher revenues. To address this problem of unfairness, authors propose a protocol where the ad click reports are encrypted by the user using the public key of the ad-network and signed by both publishers and advertisers.

In [61], authors propose CAMEO: a framework for mobile advertising that employs intelligent and proactive prefetching of advertisements. CAMEO uses context prediction, to significantly reduce the bandwidth and energy overheads, and provides a negotiation protocol that empowers applications to subsidize their data traffic costs by “bartering” their advertisement rights for access bandwidth from mobile ISPs.

In [85], authors propose a location-aware, personalized and private advertising system for mobile platforms. In this system, ads are locally broadcast to users within mobile cells. The ad matching happens locally based on the user interests. Finally ad view and click reports are collected using a DTN system. In [86], authors propose a new ad protocol that uses homomorphic and searchable encryption to allow users transmit mobile sensor data to a cloud service that responds back with the best matching contextual advertisements.

In [87], the authors present VEX, a protocol for ad exchanges to run low-latency and high-frequency ad auctions that are verifiable and auditable, in order to prevent fraud in a context where parties participating in the auction – bidders and ad exchanges – may not know each other. Based on their evaluation of the system, the authors claim that the additional storage required and latency imposed by VEX are low and practical in the context of ad auctions.

In [88] the authors present and implement PROTA, a privacy-preserving protocol for real-time advertising which uses keywords to match users interests with ads. By using bloom filters, the authors make the ad matching task efficient. The protocol relies on a trusted third party to cooperate with the ad exchange during the bidding and ad delivering phase. The authors implement and evaluate the protocol, and conclude that the time upper bond for matching ads is 200ms, which is considerable practical in the context of ad matching system.

In [89], the authors present and evaluate a system that aims at providing high-quality ad targeting in multiple scenarios, while giving the user the ability to control their privacy. The system consists of tailored extensions that mine the user behaviour locally with low overhead. The extensions generate user behavioural data that can be shared with advertisers without leaking undesirable user information. Similarly to THEMIS, the authors discuss how the system can be used by users and advertisers, and how it can be used as a replacement for the tracking-based business model in the online advertising industry.

In [90], the authors set out to formalize the concept of privacy in the context of the online advertising ecosystem and to develop a provably secure privacy-preserving protocol for the online advertising ecosystem. While the authors
claim that the definition of privacy presented in the paper is more useful compared to previous work in the online advertising context, their attempts to develop a provably secure privacy-preserving protocol has failed due to being hard to balance privacy with usefulness of the user data. The authors conjecture that cryptographic mechanisms have the potential to solve the privacy versus data usefulness conundrum. Using applying cryptography is the basis of how THEMIS proposes to preserve privacy when calculating ad rewards, providing advertisers with campaign metrics and performing confidential payments to users.

Towards a similar direction with the user rewarding schema of THEMIS, in [63] authors propose a privacy-aware framework to promote targeted advertising. In this framework, an ad broker responsible for handling ad targeting, sits between advertisers and users and provides certain amount of compensation to incentivize users to click ads that are interesting yet sensitive to them.

In [64], authors propose a targeted advertising framework which enables users to get compensated based on the amount of user tracking they sustain and the privacy they lose. The authors analyze the interaction between the different parties in the online advertising context — advertisers, the ad broker and users — and propose a framework where the interactions between the different parties is a positive-sum game. In this game, all parties are incentivized to behave according to what other parties expect, achieving an equilibrium where everyone benefits. More specifically, the users determine their click behaviour based on their interested and their privacy leakage, which in turn will influence the advertisers and ad broker to provide less invasive and better ads. THEMIS relies on a similar game theoretical approach. By providing compensation for good behaviour while providing the verification mechanisms for all parties to audit whether everyone is behaving according to the protocol, the incentives to cheat and misbehave are lower.

7 Conclusions

In this paper, we presented THEMIS: a private-by-design, trustless, and decentralized ad-platform. THEMIS (i) is auditable and transparent, it (ii) rewards the users for their interactions with ads, (iii) ensures the integrity of the billing and campaign performance reports of the advertisers, and (iv) requires zero trust from its participants.

We implemented our approach by leveraging a permissioned blockchain with smart contracts as well as zero-knowledge schemes. We measured the efficiency of our encryption scheme in the Solidity language; since our system supports batching of on-chain proof validations, it can serve more than 7M concurrent users during monthly reward payouts, while using a catalog of 128 ads. The number of concurrent users can be scaled linearly with more side-chains.

While many companies have proposed the use of blockchain for online advertising, we believe that THEMIS is the first system that delivers on that promise. Given the practicality of the approach and the combination of security, privacy, and performance properties it delivers, THEMIS can be used as a foundation of a radically new approach to online advertising.

References

[1] Panagiotis Papadopoulos, Peter Snyder, and Benjamin Livshits. Keeping out the masses: Understanding the popularity and implications of internet paywalls. In Proceedings of the The World Wide Web Conference, WWW’20. ACM, 2020.
[2] Jeff Dunn. Wikipedia is asking for donations again — here’s how much cash it already has in the bank. https://www.businessinsider.com/wikipedia-donations-profit-money-chart-2016-11, 2016.
[3] Panagiotis Papadopoulos, Panagiotis Ilia, and Evangelos P Markatos. Truth in web mining: Measuring the profitability and cost of cryptominers as a web monetization model. arXiv preprint arXiv:1806.01994, 2018.
[4] Ghassan O. Karame, Aurélien Francillon, and Srdjan Ćapkun. Pay as you browse: Micropayments as micropayments in web-based services. In Proceedings of the 20th International Conference on World Wide Web, WWW ’11, page 307–316, New York, NY, USA, 2011. Association for Computing Machinery.
[5] The Interactive Advertising Bureau (IAB) US Digital Ad Revenue Climbs to $57.9 Billion in First Half 2019, Up 17% YOY, According to IAB Internet Advertising Revenue Report. https://www.iab.com/news/us-digital-ad-revenue-climbs-to-57-9-billion-in-first-half-2019/, 2019.
[6] Muhammad Ahmad Bashir, Saifaj Adshad, William Robertson, and Christo Wilson. Tracing information flows between ad exchanges using retargeted ads. In Proceedings of the 25th USENIX Conference on Security Symposium, 2016.
[7] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. Cookie synchronization: Everything you always wanted to know but were afraid to ask. In Proceedings of The World Wide Web Conference, pages 1442–1443, 2019.
[8] N. Kshetri. The economics of click fraud. IEEE Security Privacy, 8(3):45–53, May 2010.
[9] S. Kumari, X. Yuan, J. Patterson, and H. Yu. Demystifying ad fraud. In 2017 IEEE Frontiers in Education Conference (FIE), 2017.
[10] Michael Burgi. What’s being done to rein in $7 billion in ad fraud. https://www.adweek.com/brand-marketing/whats-being-done-rein-7-billion-ad-fraud-169741/, 2016.
[11] Jessica Davies. Ghost sites, domain spoofing, fake apps: A guide to knowing your ad fraud. https://digiday.com/media/ghost-sites-domain-spoofing-fake-apps-guide-knowing-ad-fraud/, 2019.
[12] Bin Liu, Suman Nath, Ramesh Govindan, and Jie Liu. Decaf: Detecting and characterizing ad fraud in mobile apps. In Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation, NSDI’14, 2014.
[13] Apostolis Zarraxa, Alexandros Kapravelos, Gianluca Stringhini, Thorsten Hols, Christopher Kruegel, and Giovanni Vigna. The dark alleys of madison avenue: Understanding malicious advertisements. In Proceedings of the 2014 Conference on Internet Measurement Conference, IMC ’14, 2014.
[14] Jerome Dangu. Uncovering 2017’s largest malvertising operation. https://blog.confiant.com/uncovering-2017s-largest-malvertising-operation-b84cd38d6b85, 2018.
[15] Elias P Papadopoulos, Michalis Diamantaris, Panagiotis Papadopoulos, Thanasis Petsas, Sotiris Ioannidis, and Evangelos P. Markatos. The long-standing privacy debate: Mobile websites vs mobile apps. In Proceedings of the 26th International Conference on World Wide Web, 2017.
[16] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, and Christian Kreibich Phillipa Gill. Apps, trackers, privacy, and regulators. In Proceedings of the Network and Distributed System Security Symposium, NDSS’18, 2018.
[17] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P Markatos. Exclusive: How the (synced) cookie monster breached my encrypted vpn session. In Proceedings of the 11th European Workshop on Systems Security, EuroSec, 2018.
[18] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016.
[19] CXL. What is a “good” click-through rate? click-through rate benchmarks. https://cxl.com/guides/click-through-rate/
Appendices

A Smart Contracts

In this section, we specify the functionalities and properties of the smart contracts necessary to run THEMIS. In practice and at the EVM level, the policy smart contract logic and fund smart contract logic may be split into multiple smart contracts, but for the sake of simplicity, we describe the logic as part of two smart contracts per THEMIS campaign: (i) the Policy Smart Contract (Figure 9), and (ii) the Fund Smart Contract (Figure 10). We assume that both smart contracts have access to storage of the CF’s public key and the consensus participants public key (when generated).

A.1 Policy Smart Contract

Public data structures: The Policy Smart Contract (PSC) keeps its state in three public data structures: an array with the encrypted ad policies \(\mathcal{P}\), an array containing all rewards aggregates \(\mathcal{A}\) calculated by the smart contract, and an array of encrypted symmetric keys, \(\mathcal{K}\), used to encrypt the policies. The latter allows the validators to decrypt the policies and apply them to the aggregate requests.

\[\text{StorePolicy()}\]: This private function can be called only by the account which deployed the smart contract, i.e., by the Campaign Facilitator (CF). The function receives an array of uint256 types – which represent the encrypted ad policies for the campaign agreed with the advertisers (Phase 1 in 4.2.1) – and it initializes the public \(\mathcal{P}\) data structure. This function keeps its state in three public data structures: an array with all the policies and a symmetric key agreed between the CF and the corresponding advertiser.

\[\text{ComputeAggregate()}\]: It is a public function which receives the user input and the ads policies \(\mathcal{P}\) calculated by the smart contract, after having called the \(\text{StorePolicy()}\) function described above.

\[\text{Agg}\] calculated by the smart contract, after having called the \(\text{StorePolicy()}\) function described above.

\[\text{PaymentRequest()}\]: It is a public function which receives an encrypted payment request from users. Users can make use of this function with an array containing their encrypted interactions (Phase 2 in 4.2.1). The smart contract proceeds to calculate the reward aggregate based on the user input and the ads policies \(\mathcal{P}\). The aggregate is stored in the \(\mathcal{A}\) data structure, which is accessible to all users.
A.2 Fund Smart Contract

Public data structures: The Fund Smart Contract (FSC) keeps its state in multiple public data structures. The Init parameter represents whether the ad campaign has started. In order for the Init to turn to be marked as initialized (i.e., true), all campaign advertisers — which are kept in Advs — must confirm their participation by depositing the funds (to the ad campaign’s escrow account \( \mathcal{F} \)) necessary to cover their ad campaign in the smart contract account. In addition, the FSC keeps a list of all payment requests triggered by PSC and the successfully paid requests PaidReq. Finally it stores the agreed processing fees of the CF, and a value of the overall ad interaction, Aggr.Clicks, which is updated by the consensus pool.

StoreAdvID(): Private function that can be called by the CF to add new advertisers to the campaign. If the campaign has not started (i.e., Init : false), the advertisers ID is added to the Advs list. No new advertisers can be added after a campaign has started.

StoreAggrClicks(): This function has an access control policy that only a value signed by the consensus participants will update the public data structure. It is a function that is used to update the state of the smart contract overall ad interactions, Aggr.CClicks.

StoreFunds(): Public function called by the advertisers upon transferring the campaign funds to the FSC account. When all the advertisers have transferred the funds necessary to cover their ad campaign, the smart contract updates its state to Init : true.

InitialiseCampaign(): Private function that is only called by the smart contract. It sets Init : true.

SettlementRequest(): This public function has an access control policy that only a value signed by the consensus participants will update the public data structure. It is a function that is used to update the state of the smart contract overall ad interactions, Aggr.CClicks.

RefundAdvertisers(): This function is called internally by the FSC and it is triggered either when: 1) all the ads in the campaign have been 'spent' by the users or 2) when a pre-defined epoch has passed, signaling the end of the campaign. This function releases the funds to the advertisers, based on the Aggr.CClicks per advertiser in the campaign.

PayProcessingFees(): This function is - alongside with RefundAdvertisers() - called when the campaign is finished. It verifies if the CF has behaved correctly and pays the fees to the CF’s account.

RaiseComplaint(): This public function allows users to prove that the CF misbehaved. In order to call such a function, users must prove that their corresponding aggregate does not correspond to the private payment they received. To this end they disclose the aggregate value they were expecting to earn. This will prove that the CF misbehaved and the smart contract can flag the latter as such.

Policy Smart Contract

Public Storage
- Encrypted agreed policies Enc \( P \) : \[
- Aggregates Agg: \[
- Encrypted keys used to encrypt the ad policies: Enc S: \[

StorePolicy
Inputs: Encrypted policy, Enc \( P_i \).
1. Set Enc \( P_i \) = Enc \( P_i \).

StoreEncryptedKeys
Inputs: the user’s public key \( pk_{user} \), and the encrypted ad interaction, Enc.Ad.
1. Require lengthEnc \( P \) = lengthEnc \( P \).
2. Decrypt the encrypted keys
   \[
   S = \left[ \text{Dec} \left( sk_i, Enc \left( S_i \right) \right) \right]_{i=1}^N
   \]
3. Use these keys to decrypt all entries of Enc \( P \)
   \[
   P = \left[ \text{Dec} \left( S_i, Enc \left( P_i \right) \right) \right]_{i=1}^N
   \]
4. Store aggregate
   \[
   \text{Agg} \left[ pk_{user} \right] = \left[ \sum_{i=1}^N P_i \right] \cdot \text{EncVec} \left[ i \right]
   \]

GetAggregate
Inputs: the user’s public key \( pk_{user} \).
1. Return Agg \left[ pk_{user} \right]

PaymentRequest
Inputs: The user’s public key, \( pk_{user} \), decrypted aggregate, proof of decryption, and the address receiving the payment:
   \[
   E = \left[ pk_{user}, \text{Dec.Aggr.Res}, \text{SignReward}, \text{Addr} \right]
   \]
   encrypted under the validators key, Encpk \( V \).
1. Get the encrypted aggregate related to the public key,
   \[
   \text{Agg} \left[ pk_{user} \right]
   \]
2. Decrypt the encrypted payment request, to get \( E \)
3. Verify \( \text{SignReward} \) with CF’s public key, \( CF.pk \). If it validates, then:
4. Verify proof of decryption \( V = \Pi_{max}.Verif \). If \( V = \top \), append \( \text{Addr} \) to the payment request PaymentReq variable of the Fund Smart Contract.

ClaimInsufficientRefund(): This public function allows advertisers to prove that they have received insufficient refunds. To this end, advertisers simply call this function which automatically checks the validity of the claim. If the claim is valid, it flags the CF as misbehaving.

Figure 9: Description of the public storage and functionality of the Policy Smart Contract (PSC)
Fund Smart Contract

Public Storage
- Campaign initialised, Init : false
- Advertisers list, Advs : []
- Ad campaign’s escrow account, ℱ : []
- Agreed processing fees, 𝜔
- Payment request, PaymentReq : []
- Payed requests, PayedReq : []
- Overall ad interactions, Aggr.Clicks : 0

StoreAdvID
Inputs: Advertiser’s id Advs_{id} that has agreed an ad policy with CF
(1) Append to the advertisers list, the advertiser’s id Advs = Advs ∥ Advs_{id}

StoreAggrClicks
Inputs: Number of ad clicks performed by all users, Aggr.Clicks’ and signature of the value σ.
(1) Verify σ with Con.Part public key. If it validates, then
(2) Set Aggr.Clicks = Aggr.Clicks’

StoreFunds
Inputs: Fund for the campaign, ℱ_{id} which is determined by the number of impressions the advertiser wants per ad, the agreed policies and the processing fees. And the advertiser’s ID, Advs_{id}.
(1) Store fund in escrow account ℱ_{id} = ℱ_{id}
(2) If ℱ contains funds from all advertisers in Advs, call InitialiseCampaign

InitialiseCampaign
(1) Set Init = true

SettlementRequest
Input: amount, τ, and a signature of the request, σ
(1) Verify σ with CF public key, CF.pk. If it validates, then;
(2) Send τ to CF’s account.

PaymentProcessed
Input: reference of the private transaction, TxRef to address Addr
(1) Check existence of TxRef, if it exists,
(2) Add Addr to PayedReq
(3) If PayedReq == PaymentReq, then call PayProcessingFees(𝜔) and RefundAdvertisers(Advs, Aggr.Clicks)

RefundAdvertisers
Input: list of advertisers, Advs, and total value of aggregate clicks, Aggr.Clicks.
(1) If ℱ_{id} is greater than the amount ’spent’ during the campaign, refund the corresponding amount to Advs_{id}.
(2) If ℱ_{id} is smaller than the amount ’spent’, request payment to Advs_{id}.

PayProcessingFees
Input: agreed processing fees for the campaign, 𝜔, between CF and advertisers.
(1) Pay 𝜔 to CF’s account.

RaiseComplaint
Input: user’s public key, pk_{user}, reference to private payment, t_{x_priv}, and the opening of the latter, r, l, where r is the hiding value and l the payment amount.
(1) Verify that t_{x_priv} corresponds to a transaction with r, l. If yes,
(2) Fetch the payment request of the user, PaymentReq_{pk_{user}}
(3) Decrypt the request to extract the aggregate
(4) If l does not correspond to the amount fetched, flag the CF as dishonest and append the proof, i.e. the decrypted aggregate and private payment opening of the corresponding user.

ClaimInsufficientRefund
Input: advertiser’s id, Advs_{id}.
(1) Fetch the funds submitted by the advertiser, ℱ_{Advs_{id}}
(2) Fetch the number of views received by the ads of the advertisers, and multiply them by the agreed policy value, to compute the total amount spent by the advertiser
\[ Sp = \sum_{i \in S} ℱ_{i} \cdot Aggr.Clicks_{i} \]
where S is the set of all ads corresponding to advertiser Advs_{id}.
(3) If Sp plus the refund paid to the advertiser does not correspond to ℱ_{Advs_{id}}, flag the CF as dishonest.

Figure 10: Description of the public storage and functionality of the Fund Smart Contract (FSM)