Privado: Privacy-Preserving Group-based Advertising using Multiple Independent Social Network Providers

Sanaz Taheri Boshrooyeh, Alptekin Küpçü, and Öznur Özkasap

Department of Computer Engineering, Koç University, İstanbul, Turkey
{staheri14, akupcu, oozkasap}@ku.edu.tr

Online Social Networks (OSNs) offer free storage and social networking services through which users can communicate personal information with one another. The personal information of the users collected by the OSN provider comes with privacy problems when being monetized for advertising purposes. To protect user privacy, existing studies propose utilizing data encryption that immediately prevents OSNs from monetizing users data, and hence leaves secure OSNs with no convincing commercial model. To address this problem, we propose Privado as a privacy-preserving group-based advertising mechanism to be integrated into secure OSNs to re-empower monetizing ability. Privado is run by N servers, each provided by an independent provider. User privacy is protected against an active malicious adversary controlling $N-1$ providers, all the advertisers, and a large fraction of the users. We base our design on the group-based advertising notion to protect user privacy, which is not possible in the personalized variant. Our design also delivers advertising transparency; the procedure of identifying target customers is operated solely by the OSN servers without getting users and advertisers involved. We carry out experiments to examine the advertising running time under various number of servers and group sizes. We also argue about the optimum number of servers with respect to user privacy and advertising running time.

Keywords: Unlinkability, Privacy, Advertising, Online Social Networks, Privacy-Preserving Advertising, Active Adversary, Malicious Adversary.

1 Introduction

Motivation: Online social networks (OSNs) such as Facebook, Twitter, and Google+ enable social activities in the digital world. OSN servers supply storage and computational resources for the users and offer various services through which users are able to share their personal information with one another and make new friendships. Additionally, OSNs offer advertising service where the advertisers pay OSNs to find their targeted customers out of social network members. In particular, in an advertising scenario, every user fills and uploads a profile (like the Facebook profile) that contains a set of attributes varying from demographic information to personal interests and hobbies (e.g., age, education, music interests, sports activities. etc.). Likewise, advertisers submit their

requests indicating a set of attributes for the intended customers. OSN provider examines the profiles to find target customers and serves them with proper advertisements. Advertiser will be charged accordingly.

Apart from the benefits of OSNs, they come with the security problems where OSN providers monetize users' personal information by selling them to the untrustworthy advertisers [31, 35]. To address such issues, existing studies propose secure designs of OSNs using data encryption, i.e., users encrypt their data before sharing with OSNs [13, 12, 44, 45, 3, 2]. Although data encryption would mitigate the data privacy issue, it immediately disables the advertising service and cuts the significant financial benefit of advertising (the advertising revenue for Facebook in 2018 was reported as 30.83 billion dollars[1]). Due to this monetizing inability, providers would have no convincing commercial model to establish secure OSNs.

To fill this gap, we propose Privado as a privacy-preserving advertising system by which secure OSNs can efficiently perform advertising and find target customers using the privacy-protected profiles of users. Privado follows *group-based advertising* notion and also satisfies *advertising transparency* (we elaborate on these two terms in the followings). The former helps user privacy and the latter is essential for the system performance.

Personalized vs Group-based Advertising: Group based advertising is a notion that originated from PPAD [8] as a solution for the security issues that arise in personalized advertising. In fact, any secure personalized advertising ultimately compromises user privacy. That is, when an encrypted profile is matched against an advertising request (that contains a set of attributes), the success or failure of the matching indicates the presence or absence of those attributes inside that encrypted profile. Thus, after enough trials, the OSN provider can learn all the attributes inside an encrypted profile. Note that this leakage is inherent to any personalized advertising approach regardless of how the secure matching is carried out. In essence, this leakage is through the output of the matching but not the way the matching result is computed. To mitigate this privacy issue, Taheri et al. [8] introduce the notion of group-based advertising and provide a cryptographic solution for it. In group-based advertising, users are split into equal size groups during their registration and the group formation is static. Once the groups are set, every advertising request is matched against the encrypted profiles of each group. If the number of the matched profiles inside the group exceeds a threshold, the group is marked as targeted and all of its members are shown the advertisement. Note that the group matching outcome does not (and must not) indicate which profiles exactly matched the request; instead, it only asserts the total number of matches. When that number exceeds a threshold, all group members are served that advertisement. Hence, unlike the personalized advertising, the group matching result would not be linkable to the individual members.

[1] https://www.statista.com/statistics/544001/facebooks-advertising-revenue-worldwide-usa/

Advertising Transparency: In addition to user privacy, *Advertising Transparency* [8] is another requirement to be fulfilled in OSN advertising systems. That is, once the profiles of users and request of the advertiser are uploaded, the entire matching procedure is run offline by the servers without further involvement of users or advertisers. Users are served by proper advertisements (which are found via the matching procedure) in their next log-in. The advertising transparency provides benefits for system performance where the server's execution time (to match the advertisements to profiles) would not be constrained by the active time of users (who may not be online in a regular basis). As another requirement of advertising transparency, users and advertisers should not need to know or communicate with each other. This property is essential considering the fact that in an OSN like Facebook there are 2 billion monthly active users[2] and 6 million monthly active advertisers[3], and making them communicate is impractical.

Related Works: Advertising problem in secure OSNs with the aforementioned requirements results in a unique setting that has never been addressed in prior studies except [8]. In the context of secure online behavioral advertising [4, 23, 46, 16], provable user privacy and advertising transparency are lacking features. Proposals in server-aided private set intersection [30, 29, 25, 37], as well as server-aided two-party computations, cannot achieve user privacy under the coalition of server and advertiser [29, 48, 40] or they lack advertising transparency [30, 29, 25, 37]. The similar problem applies to the context of public key encryption with key search [19, 21, 49, 7, 32]. Secure Multi-party computation (SMPC) protocols [11] come with an issue where the communication complexity of parties (i.e., servers in our case) is linear with the depth of computed circuit (i.e., matching procedure). In Privado, while we utilize some SMPC techniques, we propose a very carefully designed matching function which avoids multiplication gates and the subsequent communication overhead (which can be of independent interest).

Privado vs PPAD: Taheri et al. proposed PPAD [8] as the first privacy preserving advertising proposal for secure OSNs. PPAD achieves both user privacy and advertising transparency but under two constraints. First, PPAD relies on two non-colluding servers whose cooperation compromises user privacy. Second, user privacy is guaranteed under *honest but curious* (HbC) adversarial model where all the parties must follow the protocols precisely.

In Privado, we mitigate the constraining conditions of PPAD. Firstly, we relax the assumption of PPAD about having two non-colluding servers by proposing a distributed design consisting of N servers each being operated by an independent authority. User privacy is protected even if $N-1$ servers collude. As the second enhancement on top of PPAD, Privado offers a stronger security guarantee by withstanding the malicious adversarial model where entities are allowed to deviate from the protocol specifications. In Privado, we assume that up to $N-1$ servers may maliciously collude. Also, the adversary can register fake advertising requests and user profiles.

[2] https://www.statista.com/statistics/346167/facebook-global-dau/
[3] https://www.statista.com/statistics/778191/active-facebook-advertisers/

PPAD to Privado non-triviality: A naive attempt to extend PPAD to the N server and malicious setting can be applying SMPC techniques. However, simply running SMPC between N servers would be inefficient compared to Privado. In particular, SMPC assumes the servers know the inputs, whereas in the current setting, the inputs of the servers are encrypted user profiles.

Another non-triviality lies in the fact that PPAD makes use of a privacy service provider as a trustworthy entity through which group members receive some secret information (this secret data shall be integrated into users profiles and would help in computing the group matching results). However, in Privado, none of the servers are trusted. Thus, another challenge is to perform the same confidential data dissemination using N servers which are additionally malicious. Notice that any solution to this problem should also comply with user privacy and advertising transparency.

N Server Motivation: In Privado, we utilize N servers each run by an independent provider. As long as at least one server would not collude with the rest, we protect the privacy of the users. Such setting has been similarly utilized in outsourced multi-party computations [22, 11, 1, 43]. In practice, this can be realized by having an OSN whose servers are provided by the ISPs of multiple distinct countries. As such, collusion is less likely since colluding countries would have to mutually compromise the privacy of users of their own country to their opponents. Another motivation for such a realization is that all the contributing authorities would financially benefit from such a design where they share the advertising revenue. Deploying multiple conflicting parties for the sake of privacy is similarly sought in the electronic voting (e-voting) systems [34, 50]. There, the conspiracy of tallying authorities is restrained by deploying multiple conflicting entities such as political parties of a country.

Contributions: Our contributions are abstracted as follows:

- Privado is the first privacy preserving group-based advertising for secure OSNs run by N servers managed by independent providers.
- Privado preserves user privacy under the coalition of $N-1$ servers who may additionally register an arbitrary number of advertising requests as well as control $k-2$ users within each group, where k is the group size.
- Privado withstands a malicious adversarial model where the corrupted parties do not conform to the protocol descriptions.
- We prove the security of Privado formally employing the game-based security definition in PPAD [8].
- We also carry out experiments to examine Privado's advertising running time under various number of servers and group sizes. Additionally, we argue about the optimum number of servers with respect to user privacy and advertising running time.

The rest of paper is organized as follows. We present the system model, security objectives and the adversarial model of Privado in Section 2. Preliminaries and definitions are provided in Section 3 followed by the full construction in Section 4. Section 5 is dedicated to the asymptotic and concrete performance of Privado and the comparison with PPAD. The formal security definition and

proof come in Section 6. Section 7 overviews prior studies and highlights their shortcomings. Concluding remarks are given in Section 8.

2 System Model

2.1 Model

Privado is comprised of *users*, *advertisers*, and N *servers* denoted by S_1,...,S_N. The system overview is illustrated in Figure 1. N servers jointly create an encryption key pk and distribute the corresponding decryption key shares $dk_1, ...dk_N$ among themselves. Hence, performing decryption requires all the servers to contribute using their decryption key shares. Servers also decide on the group size and threshold value to be used in the group matching procedure.

Users fill in a profile (similar to Facebook profile) in which they provide their preferences as a list of attributes varying from demographic information to personal interests and hobbies (e.g., age, education, music interests, sports activities, etc.). Then, the user encrypts his profile under pk and hands it over to all the servers. On the other side, advertisers create their advertising requests, which include the attributes they seek in their target customers, e.g., {painting $\wedge$ football $\wedge$ computer engineer}. Requests are then submitted to each server in the plaintext form. Each server has a local database in which it stores all the advertising requests and encrypted user profiles registered in the system.

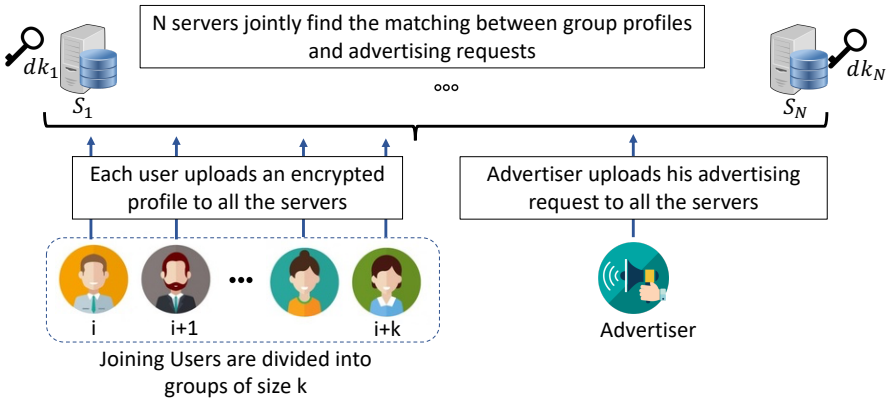

Fig. 1. Privado System Overview. Users encrypt their profiles using an encryption key whose corresponding decryption key is split among the N servers as $dk_1, ..., dk_N$.

Privado applies the group-based advertising notion. For that, users are randomly divided into groups of identical size at their arrival time. As discussed earlier, the grouping of users is essential for user privacy and should not be

done based on similar interests. For every advertising request, N servers jointly examine all the groups and identify the target ones: i.e., groups whose number of matched profiles exceeds a system-wise threshold. All the members of a target group are served by the advertisement (since advertising to a subset would violate user privacy). The non-target groups are skipped. Subsequently, the advertiser is charged based on the number of target groups (hence users) to which the advertisement is shown. We remark that group-based advertising degrades advertising accuracy (losing some target customers) where we do not advertise to the groups with insufficient target members. Also, there would be some non-target users who receive irrelevant advertisements since they belong to a target group. Hence, this is the cost of achieving user privacy. We refer the interested readers to PPAD [8] for an extensive analysis of the advertising accuracy in group-based advertising over various group sizes and threshold values.

2.2 Security Goal

The security goal of Privado is to protect user privacy. User privacy indicates the inability of any adversarial entity to bind the group matching result to a particular group member, as defined in PPAD [8]. This means that for a target group, none of the servers would be able to conclude whether a particular group member has the attributes of the advertising request or not. However, it is important to realize that any advertising system is inherently prone to some implicit privacy leakage regardless of how secure it is designed. That is, once a group is qualified/disqualified as a target for an advertising request, this reveals the inclusion or exclusion of the queried attributes among the group members. This leakage is inevitable even though the matching occurs entirely on the encrypted profiles. In personalized advertising, this leakage immediately breaks user privacy [29, 48, 30, 29, 25, 37] whereas in the group-based counterpart the inclusion or exclusion is not attributable to a particular group member. PPAD supplies a formal security definition for user privacy that we present in Section 6.1. We utilize this definition to formally prove the security of Privado.

2.3 Adversarial Model

Privado considers the adversarial model where an adversary may corrupt any subset of $N-1$ servers, create arbitrarily many advertising requests and fake $k-2$ profiles per group (k is the group size). Notice that, having more than $k-1$ fake profiles in a group converts the group-based advertising to the personalized variant (adversary knows the content of $k-1$ profiles hence can immediately link the group-matching results to the unknown profile), which, as discussed earlier, violates user privacy. We consider **static corruption**, i.e., the adversary selects which party to corrupt before the protocol starts. Corrupted parties are **active/malicious adversaries** who may refuse to follow the protocols' specifications. We assume parties communicate through **insecure but authenticated channels**.

3 Definitions and Preliminaries

3.1 Notation

We write $x \leftarrow X$ to denote picking an element x uniformly at random from set X. $a||b$ represents concatenation of a with b. $|A|$ stands for the number of elements in set A. $\boxed{a}$ represents a ciphertext embodying the value a. Likewise, $\boxed{B}$ where B is a vector, means the element-wise encryption of B. For shorthand, we use PPT for a probabilistic polynomial time entity. We write $Enc_{pk}(m, r)$ to denote encryption of a message m using an encryption key pk and a randomness r.

3.2 Definitions

Negligible: A function f is called negligible if for all positive polynomials p, there exists a constant C such that for every value $c > C$ it holds that $f(c) < \frac{1}{p(c)}$.

Secure Multi-Party Computation: Consider the ideal functionality $F(in_1, ..., in_N) = (f_1(in_1, ..., in_N), ..., f_N(in_1, ..., in_N))$ running by a trusted third party that receives inputs (in_i) from i^{th} party and delivers $f_i(in_1, ..., in_N)$. Let γ^F be a multi-party protocol to compute F. γ^F is said to securely realize F if for every PPT adversary A with auxiliary input $aux \in \{0,1\}*$ attacking protocol γ^F, there exists a PPT simulator Sim for the ideal functionality F, that $\forall$ security parameter λ:

$$\{IDEAL_{F,Sim(aux),P_c}(in_1, ..., in_N, \lambda)\}\} \equiv_c \{REAL_{\gamma^F,A(aux),P_c}(in_1, ..., in_N, \lambda)\} \quad (1)$$

The left and right side of this equality represent the output of parties in interaction with F and γ^F, respectively. P_c is the set of corrupted parties controlled by the adversary/simulator. $\equiv_c$ stands for computational indistinguishability.

Hybrid Model: Let γ^F be a multi-party protocol that securely realizes ideal functionality F and assume θ is another protocol that makes use of γ^F as a sub-protocol. In the hybrid model, security of θ can be proven by replacing γ^F with its ideal functionality F (as if there is a trusted third party running F). This would be called F-hybrid model [17].

3.3 Preliminaries

Bloom Filter [6] is a data structure for the set representation which also supports insertion and membership check queries. A Bloom filter is formed as a bit array of size p, and d hash functions denoted by $H_1(.), ..., H_d(.)$. To insert an element x to the Bloom filter, all the hash functions are evaluated on x; i.e. $H_1(x)..., H_d(x)$. The output of the hash functions indicate the indices of the bit array that shall be set to 1. To check the membership of an element y, similarly the hash functions are evaluated on y which results in d indices. If all the

corresponding bit values are 1, then y counts as an element of the set with the probability of $1-(1-((1-\frac{1}{p})^{d})^{e})^{d}$. e is the total number of elements inserted into the Bloom filter. Otherwise (if at least one of the checked indices is 0), y is definitely not a member. Throughout the paper, we refer to the creation of a Bloom filter with $BFCreate(Att)$ where Att is the set of elements to be inserted.

Super-Increasing Set: A super-increasing set $A = \{a_1, ...a_k\}$ of size k is a set of k positive real numbers such that each element of the set is greater than the aggregate of its preceding elements in the set [42]. That is,

$$\forall i, a_i > \sum_{j=1:i-1} a_j \tag{2}$$

IND-CPA Encryption Scheme: For a public key encryption scheme $E = (Gen, Enc, Dec)$ we define the IND-CPA game $PubK^{CPA}_{A,E}(\lambda)$ as [27]

$$\begin{aligned}&(pk, dk) \leftarrow Gen(1^{\lambda}); (M_0 = \{m_{0,i}\}_{i=1:L}, M_1 = \{m_{1,i}\}_{i=1:L}, history) \leftarrow A(pk)\\ &\text{s.t. } |m_{0,i}|_{i=1:L} = |m_{1,i}|_{i=1:L}; b \leftarrow \{0,1\}; C = \{c_i \leftarrow Enc(Pk, m_{b,i})\}_{i=1:L};\\ &b' \leftarrow A(history, C) : \text{output is 1 if } b == b'\end{aligned} \tag{3}$$

Encryption E is IND-CPA if for every probabilistic polynomial time adversary A, there exists a negligible function $negl(\lambda)$ s.t.

$$Pr[PubK^{CPA}_{A,E}(\lambda) = 1] < \frac{1}{2} + negl(\lambda) \tag{4}$$

or equivalently [27],

$$|Pr[output(PubK^{CPA}_{A,E}(\lambda, 0)) = 0] - Pr[output(PubK^{CPA}_{A,E}(\lambda, 1)) = 0]| \tag{5}$$

where $PubK^{CPA}_{A,E}(\lambda, b)$ indicates the output of IND-CPA experiment when the challenge bit is b.

(N,N)-Threshold Additive Homomorphic Encryption Scheme with a Distributed Key Generation: An additive homomorphic encryption scheme is a public key encryption scheme with key generation Gen, encryption Enc, and decryption Dec algorithms which additionally enables computation over plaintexts using ciphertexts. That is, for any two messages m_0, m_1 (from the message space) encrypted as $c_0 = Enc_{pk}(m_0), c_1 = Enc_{pk}(m_1)$, one can compute the summation of messages in the encrypted format as $Enc_{pk}(m_0+m_1) = c_0 \odot c_1$ where pk is the encryption key and $\odot$ is the homomorphic operation over the ciphertext.

An example is Paillier encryption in which multiplication of the ciphertexts results in the summation of the underlying plaintexts i.e., $c_0 \cdot c_1 = Enc_{pk}(m_0 + m_1)$. In the (N, N)-threshold homomorphic encryption, the decryption key is distributed among N parties such that the presence of all of them (N out of N) is required to make correct decryption. Generation of the key in the threshold settings usually relies on a trusted party who creates and distributes the decryption key shares and then leaves the system. However,

in the present work, we leverage a distributed key generation protocol proposed by [18] for the Paillier setting. We refer to that encryption scheme by $TEnc = (DKeyGen, Dsk, Enc, DDec)$ whose details come next.

1. $pk \leftarrow DKeyGen(1^\lambda)$ is a distributed protocol run by N parties $S_1, ..., S_N$ to compute the public key pk as a composite modulus N with an unknown prime factorization $q_1.q_2$.
2. $dk_1, ...dk_N \leftarrow Dsk(pk)$ is a distributed protocol run by $S_1, ...S_N$ to generate decryption key shares of the given public key pk. Each party S_i receives one share i.e., dk_i.
3. $C \leftarrow Enc(N, m)$ is the encryption algorithm to convert the plaintext to a random ciphertext. A random number r is chosen from Z_N^*. The ciphertext is set to $C = g^m \cdot r^N \bmod N^2$ where $g = 1 + N$.
   Additionally, Paillier encryption comes with re-encryption algorithm to re-randomize a given ciphertext C i.e., $C' \leftarrow ReEnc(C, r') = C \cdot r'^N \bmod N^2$ where $r' \in Z_N^*$. We may eliminate r' and write $ReEnc(C)$ for shorthand.
4. $m = DDec(dk_1, ..., dk_N, C)$ is a distributed protocol in which all the parties $S_1, ..., S_N$ contribute their respective shares of the decryption key i.e., $dk_1, ..., dk_N$ to decrypt a ciphertext C to the plaintext m.

We instantiate an (N,N)-threshold encryption scheme $TEnc$ using the proposal of [18]. We present the security guarantees of $TEnc$ as an ideal functionality F_{THRESH} as shown in Figure 2 (generalizing two-party definitions of [18]).

Zero-Knowledge Proof (ZKP) of Knowledge: is a proof system $< P, V >$ for a language L defined over relation R i.e., $L = \{x | \exists\, \omega : (x, \omega) \in R\}$ by which a prover P knowing witness ω can prove the validity of a statement i.e., $x \in L$ to a verifier V. Let $(P(\omega), V(z, r))(x)$ be the output of V (namely, 1 if V accepts the proof, 0 otherwise) in interaction with P upon the common public statement x. The verifier holds the auxiliary input z and the random tape r whereas P owns the private witness ω. A zero-knowledge proof system satisfies three properties which are abstracted as follows [15]:

- Perfect Completeness: an honest prover can always convince the honest verifier on a valid statement $x \in L$. In principle, for every $(x, \omega) \in R$

$$Pr[(P(\omega), V)(x) = 1] = 1 \tag{6}$$

- Computational Soundness: A dishonest prover is unable to make a valid proof for an invalid statement $x \notin L$ unless with a low probability. That is, $\forall (x, \omega) \notin R$ and for all dishonest PPT prover P^*,

$$Pr[(P^*(\omega), V)(x) = 1] = 1 - 2^{-t} \tag{7}$$

  2^{-t} is called soundness error and can get arbitrarily small for the large values of t.
- Computational Zero-knowledge: The proof system does not reveal anything beyond the correctness of the statement $x \in L$. More formally, a proof system

$\mathbf{F}_{THRES}$

Key Generation: Upon receiving request $(Generate, 1^\lambda)$ from party P_i $(i \in [1, N])$, F_{THRES} records $(P_i, Generate, 1^\lambda)$ in the database and sends it to the adversary. Upon the receipt of request $(P_i, Generate, 1^\lambda)$ from all the N parties $(\forall i \in [1, N])$, F_{THRES} sends (RandInput) to the adversary and receives (GenInput, r). Then, F_{THRES} uses its own randomness together with r to generate an encryption key pk and its corresponding decryption key dk. F_{THRES} records dk and outputs pk to the adversary. If the adversary responds with *continue* then F_{THRES} delivers pk to all the other parties and ignores any message of this form, otherwise, sends *abort* to all the other parties.

Decryption: Once $(Decrypt, c)$ message is received from some party P_i $(i \in [1, N])$, F_{THRES} operates as below:

1. If no key was created, F_{THRES} ignores the request.
2. If a key was already created, then F_{THRES} records $(P_i, Decrypt, c)$ and sends it to the adversary. Once requests for the decryption of c are received from all parties $P_{i=1:N}$ then F_{THRES} sends $(Decrypt, c)$ to the adversary. Adversary responds with the receiver set $RC \subseteq \{1, ..., N\}$. If RC is empty, F_{THRES} sends *abort* to all the parties. Otherwise, F_{THRES} decrypts the ciphertext c as $Dec_{dk}(c)$ and sends the result to the parties specified in RC.

Fig. 2. The ideal functionality F_{THRES} for the distributed (N, N)-threshold encryption scheme $TEnc = (DKeyGen, Dsk, Enc, DDec)$. F_{THRES} captures the security properties of Key Generation (i.e., $DKeyGen$ and Dsk) and Decryption ($DDec$) which are multi-party protocols. However, Enc is a single party algorithm (not a multi-party protocol) and thus is not presented as a functionality.

(P, V) is computational Zero-knowledge if there exists a PPT simulator Sim s.t. for every PPT verifier V^* we have

$$\{(P(w), V^*(z, r))(x)\} \stackrel{c}{\equiv} \{(Sim^{V^*(x,z,r,.))}\} \tag{8}$$

$Sim^{V^*(x,z,r,.)}$ indicates the output of simulator with oracle access to $V^*(x, z, r, .)$.

We refer to an *interactive proof system* as the proof that is an interactive two-party protocol run between the polynomial-time prover and the verifier. On the other hand, in a *non-Interactive proof*, the proof generated by the prover can be verified without further interaction with the prover.

Non-Interactive Zero-Knowledge Proof of Plaintext Range (NI-POPR): This is a proof system to prove that a ciphertext $\boxed{m}$ encrypts a value of a particular range *Range* i.e., $m \in Range$. We illustrate a non-interactive version of such POPR with NI-POPR($\boxed{m}, R$).

We present functionality F^R_{POPR}, as shown in Figure 3, to capture the security requirements of an ideal $POPR$ protocol.

F^R_{POPR} **with a relation** R

- F_{POPR-R} receives (Prove, P, V, (pk, $\boxed{m}$, *Range*), ω) from prover P and hands over it to the adversary.
- If the adversary responds with *abort*, then F^R_{POPR} sends *abort* to the verifier V.
- If the adversary responds with *continue*, If ((pk, $\boxed{m}$, *Range*), ω)$\in R$, F^R_{POPR} sends (proof, P, V, (pk, $\boxed{m}$, *Range*)) to the verifier V. Otherwise, outputs (disproof, P, V, (pk, $\boxed{m}$, *Range*)) to V.

Fig. 3. The ideal functionality F^R_{POPR} for proof of plaintext range.

In section 4, we will make use of $POPR$ for ciphertexts embodying 0 or 1 values. Hence, we deploy the proof system proposed by [39] over the following relation R (Equation 9). μ belongs to Z^*_N:

$$R = \{((pk, \boxed{m}, Range), \omega =< m, \mu >) |\ m \in Range \wedge \boxed{m} = Enc_{pk}(m, \mu)\} \tag{9}$$

Non-Interactive Zero-Knowledge Proof of Correct Multiplication (NI-POCM): The purpose of POCM is to prove that a ciphertext is the correct multiplication of two given ciphertexts under a given public key pk. Namely, $\boxed{c} = \boxed{a * b}$ where $\boxed{a}$ $\boxed{b}$ and $\boxed{c}$ are given. We illustrate a non-interactive version of such POCM with NI-POCM($\boxed{a}, \boxed{b}, \boxed{c}$). The ideal function F^R_{POCM}, presented in Figure 4, captures soundness, completeness and zero-knowledge properties of a secure $POCM$ protocol.

F^R_{POCM} **with a relation** R

- F^R_{POCM} receives (Prove, P, V, (pk, $\boxed{a}$, $\boxed{b}$, $\boxed{c}$), ω) from prover P and hands over it to the adversary.
- If the adversary responds with *abort*, then F^R_{POCM} sends *abort* to the verifier V.
- If the adversary responds with *continue*, if ((pk, $\boxed{a}$, $\boxed{b}$, $\boxed{c}$), ω)$\in R$, F^R_{POCM} sends (proof, P, V, (pk, $\boxed{a}$, $\boxed{b}$, $\boxed{c}$)) to the verifier V. Otherwise, outputs (disproof, P, V, (pk, $\boxed{a}$, $\boxed{b}$, $\boxed{c}$)) to V.

Fig. 4. The ideal functionality F^R_{POCM} for the proof of correct multiplication.

We base our POCM on the technique presented in [11] for Paillier encryption setting. The corresponding relation R is formulated in equation 10. μ and γ are elements of Z^*_N.

$$R = \{(pk, \boxed{a}, \boxed{b}, \boxed{c}), \omega =< a, \mu, \gamma >) | \boxed{a} = Enc_{pk}(a, \mu) \wedge \boxed{c} = ReEnc(\boxed{b}^a, \gamma)\} \tag{10}$$

We further leverage Fiat-Shamir method [14] to achieve perfect zero-knowledge and a non-interactive POCM in random oracle model.

Mix Network: Mix network (mix-net) is a multiparty system which converts a set of input data to an untraceable output [41]. The input is usually a set of ciphertexts i.e., $\overleftarrow{C}_{in} = \{C_1, ..., C_k\}$ and the output is the re-encrypted and permuted version of the input i.e., $\overleftarrow{C}_{out} = \{C'_{\pi(1)}, ..., C'_{\pi(k)}\}$ where π is a permutation and $C'_{\pi(i)}$ denotes the re-encryption of $\pi(i)^{th}$ element of C_{in}. A mix-net comprises multiple servers (mixers) $\{S_1, ..., S_N\}$ where each S_i in turn computes a randomly permuted and re-encrypted output C_{out_i} from $C_{out_{i-1}}$.

Verifiable Shuffles are used to implement mixers. A verifiable shuffle VS is a tuple $VS = (E, SH, (P, V))$ where $E = (Gen, Enc, Dec, ReEnc)$ is an encryption scheme with key generation Gen, encryption Enc, decryption Dec and re-encryption $ReEnc$ algorithms. SH denotes shuffle algorithm whose input is a set of ciphertexts i.e., $\overleftarrow{C}_{in} = \{\boxed{m_1}, ..., \boxed{m_k}\}$ and the output is the re-encrypted and permuted version of the input i.e., $\overleftarrow{C}_{out} = \{\boxed{m_{\pi(1)}}, ..., \boxed{m_{\pi(k)}}\}$. (P, V) is a proof system used to prove that there exists a permutation π and some randomnesses which can covert the input ciphers to the output ciphers.

A verifiable shuffle should satisfy the following properties: 1) *shuffle privacy* that is the permutation must remain secret to any outsider (this features usually relies on the IND-CPA security of the underlying encryption E) and 2) *shuffle verifiability* which means the correct construction of the output should be verifiable (that relies on the soundness of the proof system). The shuffle verifiability guarantees the robustness of a mix-net even when some number of mixers are corrupted. We define the ideal functionality F^R_{VS} (Figure 5) to capture the security

properties of a verifiable shuffle proof system. $\overleftarrow{C}_{out}$ is the correct permutation of $\overleftarrow{C}_{in}$ if the prover P knows a witness ω for which $(\overleftarrow{C}_{in}, \overleftarrow{C}_{out}, w) \in R$.

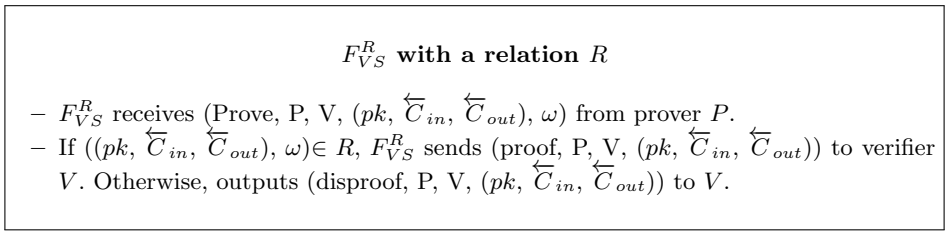

Fig. 5. The ideal functionality F_{VS}^{R} for the proof system of a verifiable shuffle scheme.

We employ the Pallier-based mix-net protocol proposed by [38]. Their underlying shuffle scheme has an interactive proof system which is transformable to a non-interactive version using Fiat-Shamir method [14] in random oracle model. R is defined as in Equation 11. π is the permutation, $\overleftarrow{C}_{in} = \{\boxed{m_1}, ..., \boxed{m_k}\}$ and $\overleftarrow{C}_{out} = \{C_1, ..., C_k\}$.

$$R = \{(pk, \overleftarrow{C}_{in}, \overleftarrow{C}_{out}), \omega =< \pi, \{\mu_i\}_{i=1}^{k} >) \mid \forall C_i \in \overleftarrow{C}_{out},\ C_i = ReEnc(\boxed{m_{\pi(i)}}, \mu_i)\} \tag{11}$$

4 Privado

4.1 Design Challenges

In this section, we list some possible threats and attacks that can be attempted by a malicious adversary in our system model. We also sketch our proposed solution next to each item. The attacks are not limited to this list; our goal is to highlight some of our design difficulties as well as provide an intuition of our design choices.

In Section 6, we present a formal security definition capturing user privacy followed by a concrete security proof of Privado. Our formal proof does not impose any strategy on the adversary and considers a black box adversary. This implies that our advertising system defeats any misbehavior of adversary including the ones listed below.

Profile Replay Attack: Profile replay attack refers to the situation where a malicious server attempts registering a (corrupted) user inside a group employing the profile of an honest member of that group. That is, the corrupted server re-encrypts the honest user's profile and submits it as a new one. Note that due to the IND-CPA security of the encryption scheme, the honest servers would not notice such profile duplication. This duplication influences the pattern in the

group matching result (i.e., the total number of matched users in each group). For clarification, assume a group of size k with 2 honest members and $k-2$ dishonest ones. Let P_1 and P_2 denote the profiles of honest members. One of the corrupted group members submits a duplicate of P_1 and the rest of $k-3$ corrupted members leave their profiles empty (without any attributes). Therefore, the matching result of P_1 is always counted twice in the group matching result. Thus, if P_1 matches an advertising request then the total number of target users would be a value larger than 2, otherwise not. This indeed enables a corrupted server to link the result of matching to a particular member (the genuine holder of P_1 in the above example).

Privado defeats this attack by requiring each user to prove in zero-knowledge that he knows the content of the submitted encrypted profile. Consequently, the adversary cannot make valid proof of an encrypted profile with unknown content. The proof includes *proof of plaintext range* as well as *proof of correct multiplication*. We supply the details in Section 4.3.

Compound Group Matching: Compound group matching attack occurs when a corrupted server does not commit to the initial grouping of users, i.e., for each advertising request, it groups profiles in an arbitrary way. This enables an adversary to deliberately group a victim profile with multiple different groups and according to the changes in the matching results (before and after inclusion of the victim profile) learns which advertising requests match the profile.

Privado stands this attack by employing an (N, N)-threshold encryption scheme whose decryption key is divided among N servers each run by an independent provider. Servers are all aware of the initial grouping of the users. Also, at least one of the servers is non-colluding by assumption, i.e., would not contribute its decryption power with other servers for fake groups. Thus, compound group matching, which relies on the decryption operation, would be impossible.

Servers Equivocation: This attack refers to any deviation of the servers from the execution of instructed protocols. Our approach stands against the attack using replicated computations. That is, multiple servers run an identical set of computations and they shall end up with the same local results. In the case of inconsistency, an equivocation is detected. We make use of additive homomorphic encryption (which is secure against malicious entities) to devise the servers computations.

Privado vs PPAD: Technically, because we have a harder problem compared to PPAD [8], our solution employs many other tools, including threshold additive homomorphic encryption, zero knowledge proofs of knowledge, verifiable shuffles, and mix networks.

4.2 Construction Overview

In this part, we provide an overview of Privado's protocols and their objectives. Protocols are **Initialization**, **User Registration**, **Advertiser Registration**, and **Advertising**. The detailed construction is provided in section 4.3.

Initialization: The system life-cycle starts by servers running the initialization protocol to set up the necessary protocol parameters. This includes a thresh-

old homomorphic encryption scheme whose decryption key is shared among the servers. The key generation is a distributed protocol without making use of any trusted third party. Each server also utilizes a database to keep a copy of every profile and advertising request.

User Registration: A user joins the system by executing the user registration protocol. Each user is assigned to a particular group at his arrival time and obtains a group membership identifier (MID) that uniquely identifies him among his group-mates. The membership ID assignment should be at random and private; otherwise user privacy can be violated (see Section 6). Servers distribute MIDs among the group members privately and randomly by the help of a mix-net protocol. MIDs shall help in the calculation of the group matching results. Upon the receipt of MID, the user creates his encrypted profile. Each profile is comprised of a set of attributes which are modeled by a Bloom filter. The final encrypted profile comprises element-wise encryption of the Bloom filter (in which the MID is also integrated). Additionally, the user proves in zero knowledge that the profile was constructed properly.

Advertiser Registration: Similar to the profile, advertising request consists of a set of attributes that shall be converted to a Bloom filter format. The advertiser submits his request to all the servers. The servers insert the request into their local databases.

Advertising: Servers run advertising protocol to determine the target groups for a given advertising request. Recall that a target group is the one in which the total number of matched profiles (i.e., target users) exceeds a system-wide threshold. In a nutshell, the advertising protocol consists of two parts: *Aggregation* and *Matching*. During the aggregation phase, each server locally computes the matching results of individual members in the encrypted format and then aggregates them into a single ciphertext. Aggregation helps break any linkability between the matching results and individual group members. Next, the servers collaborate to decrypt the aggregate by running a threshold decryption protocol. Each server de-aggregates the aggregate value (that is now in plaintext) to identify the total number of profiles that match to the advertising request.

4.3 Full Construction

This section presents the detailed construction of Privado. We assume that the servers have synchronized states. However, the synchronization of servers does not serve any privacy purpose, that is, the lack of synchronization would not cause any privacy issue.

Initialization: Initialization protocol is run by N servers to set the system parameters as follows.

- Servers initially agree on protocol parameters and publish them publicly. This includes the group size k, the threshold value Thr which is used for the group matching, the Bloom filter size p and its hash functions. Also, servers

construct a super-increasing set $\Delta = \{\delta_1, ..., \delta_k\}$ of size k called *Membership Identifier Set* whose elements satisfy Equation 12.

$$\forall m \in \{1, ..., k\}, \delta_m > \sum_{i=1}^{m-1} \delta_i * p \tag{12}$$

where p is the size of the Bloom filter. Elements of Δ are used in the user registration protocol.

- Servers jointly establish a distributed (N,N)-threshold additive homomorphic encryption scheme $TEnc = (DKeyGen, Dsk, Enc, DDec)$. As such, servers run $DKeyGen$ to generate an encryption key pk which shall be publicized to the whole system. Servers engage in the execution of Dsk protocol to create the shares of the corresponding decryption key. As a result, every (i^{th}) server obtains an additive share of the decryption key ($dk_{i=1:N}$) which keeps it private. Correct decryption requires all servers contributing their decryption key shares. Henceforth, for shorthand, we write Enc to denote encryption under pk.
- One of the servers, namely S_j, encrypts Δ as $\boxed{\Delta} = (\boxed{\delta_1} = Enc(\delta_1, r_1), ..., \boxed{\delta_k} = Enc(\delta_k, r_k))$ using the randomnesses $r_{i=1:k}$. S_j communicates $\boxed{\Delta}$ together with the randomnesses $r_{i=1:k}$ to all the other severs $S_{i=1:N, i\neq j}$. Each server $S_{i=1:N}$ recomputes the encryptions, namely, for $i \in [1, k]$ computes $Enc(\delta_i, r_i)$ and compares against $\boxed{\delta_i} \in \Delta$. Servers abort in the case of mismatch, otherwise, store $\boxed{\Delta}$ in their local databases. As we will present in the user registration part, $\boxed{\Delta}$ shall be used as the input to the mix-net.

User Registration: Figure 6 depicts user registration protocol by which the user registers his profile to N servers. User and servers interact through an authenticated channel as given below.
Servers:

- *Group assignment:* Servers decide on the user's group identifier i.e., GID which determines the group the user belongs to (step 1.1 of Figure 6). The group assignment must be at random and can rely on users arrival order. Servers assign GIDs to the users incrementally. Namely, servers assign $GID = 1$ to the first set of k registered users, and $GID = 2$ for the second set of k registered users and so on. For this sake, each server keeps track of the number of joining users.
- *Membership ID assignment:* Next, servers assign a private integer called *membership identifier* (for shorthand, membership ID denoted by δ) to each user that uniquely identifies him inside his group (step 2 of Figure 6). Since the uniqueness of δs must be preserved within each group, servers only create one set of identifiers i.e., Δ in the initialization phase, and keep assigning the same identifiers for every group but under different permutations.

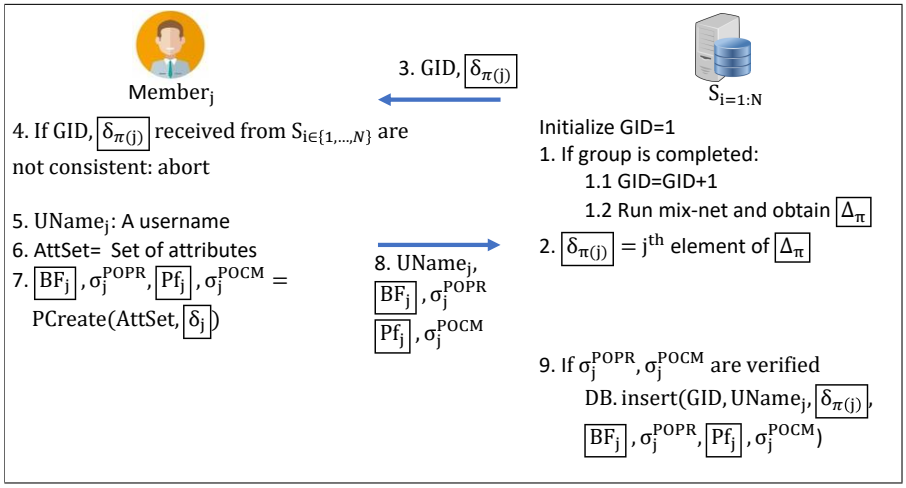

Fig. 6. An instance of User Registration protocol (UReg) between jth group member ($j \in [1, k]$) and ith server ($i = 1 : N$). When a group of profiles is registered (step 1), servers increment the GID as well as jointly run mix-net to make a new shuffle of the membership identifier set $\boxed{\Delta}$. The user obtains the group information (at step 3) from all the servers and delivers the profile (at step 8) to all of them.

For every group of k users, servers shuffle the encrypted membership identifier set i.e., $\boxed{\Delta}$ ($\boxed{\Delta}$ is generated in the initialization phase) through a mix-net execution (step 1.2 of Figure 6). We remark that **mix-net is only run once for each group**. Let $\boxed{\Delta_\pi} = (\boxed{\delta_{\pi(1)}}, \ldots, \boxed{\delta_{\pi(k)}})$ be the output of mix-net where π indicates the permutation. Each server $S_{i=1:N}$ stores $\boxed{\Delta_\pi}$ in its local database. For the jth member of the group, each server $S_{i=1:N}$ delivers the jth element of $\boxed{\Delta_\pi}$ (i.e., $\boxed{\delta_{\pi(j)}}$) to that user (step 2 of Figure 6). Note that since mix-net preserves shuffle privacy, none of the servers knows the permutation π and hence does not know which δ is given to which group member. Thus, the private assignment of membership identifier is satisfied. Additionally, shuffle verifiability of mix-net guarantees that $\boxed{\Delta_\pi}$ is the correct permutation of $\boxed{\Delta}$ (even in the presence of $N-1$ corrupted servers).

- Each server $S_{i=1:N}$ sends GID and the membership identifier $\boxed{\delta_{\pi(j)}}$ to the user (step 3 of Figure 6).

User:

- The user obtains GID as well as $\boxed{\delta_{\pi(j)}}$ from each server $S_{i=1:N}$ (step 3 of Figure 6). If the GID and the ciphertext $\boxed{\delta_{\pi(j)}}$ sent by all the servers are identical (step 4 of Figure 6), user continues to the next step. Otherwise, an equivocation from the server side is detected and user aborts.

Algorithm 1 PCreate(AttSet, $\boxed{\delta_{\pi(j)}}$)

1: $BF_j = \{b_{j,i}\}_{i=1:p} = BFCreate(AttSet)$
2: $\boxed{Pf_j} = \{\boxed{Pf_{j,i}}\} = \{ReEnc(b_{j,i} * \boxed{\delta_{\pi(j)}})\}_{i=1:p}$
3: $\boxed{BF_j} = \{\boxed{b_{j,i}}\}_{i=1:p} = \{Enc(b_{j,i})\}_{i=1:p}$
4: $\sigma_j^{POPR} = \{\text{NI-POPR}(\boxed{b_{j,i}}, \{0,1\})\}_{i=1:p}$
5: $\sigma_j^{POCM} = \{\text{NI-POCM}(\boxed{b_{j,i}}, \boxed{\delta_{\pi(j)}}, \boxed{Pf_{j,i}})\}_{i=1:p}$
6: **return** $\boxed{BF_j}, \sigma_j^{POPR}, \boxed{Pf_j}, \sigma_j^{POCM}$

- *Profile creation and submission:* The procedure of profile creation is shown in Algorithm.1. The user inserts his set of attributes *AttSet* into a Bloom filter data structure (line 1 of Algorithm.1). Then, the user multiplies each element of BF with $\boxed{\delta_{\pi(j)}}$ and re-encrypts the result (line 2 of Algorithm.1). The resultant vector $\boxed{Pf_j}$ constitutes the profile of user. Additionally, the user must create a proof asserting that the profile is well-formed. Namely, the element of $\boxed{Pf_j}$ are either encryption of zero or the assigned $\boxed{\delta_{\pi(j)}}$. As such, user performs the following.
  1. The user encrypts his Bloom filter (line 3 of Algorithm.1). Then, for every element of $\boxed{BF_j}$, he creates a proof of plaintext range $[0, 1]$ (line 4 of Algorithm.1).
  2. Using proof of correct multiplication, the user proves that every element of $\boxed{Pf_j}$ is the correct multiplication of the corresponding element in $\boxed{BF_j}$ with $\boxed{\delta_{\pi(j)}}$ i.e., $\boxed{Pf_{j,i}} = \boxed{b_{j,i} \cdot \delta_{\pi(j)}}$ (line 5 of Algorithm.1).
- Finally, the user submits $\boxed{BF_j}, \boxed{Pf_j}$ together with the proofs $\sigma_j^{POPR}, \sigma_j^{POCM}$ to all servers (steps 7-8 of Figure 6).

Servers:

– Each server $S_{i=1:N}$ verifies the proofs $\sigma_j^{POPR}, \sigma_j^{POCM}$ and accepts or rejects accordingly (step 9 of Figure 6). If the verification is successful, servers insert the profile together with its GID, and $\boxed{\delta_{\pi(j)}}$ into their local databases.

Advertisement Registration: This protocol, given in Figure 7, is run between the advertiser and all the servers to register an advertising request to the system. They interact as follows.
Advertiser:

- Advertiser creates a Bloom filter from the set of attributes (denoted by $TAud$) he seeks in his target users (steps 1-3). The target audience of the advertising request are the profiles which contain the conjunction of attributes

in $TAud$. Thus, if an advertiser wants to find users with attributes X **OR** Y, he must split the request and submit it as two separate requests: one for X and the other for Y.
- Let $Req = \{r_1, r_2, ..., r_p\}$ be the created Bloom filter (p is the size of Bloom filter). The advertiser hands over Req as well as the product advertisement (denoted by $Product$) to each server $S_{i=1:N}$ (step 4).

Servers:

- Servers assign a request number RID to the advertising request (step 5) and return it to the advertiser (step 7). RIDs are assigned incrementally, thus, servers keep track of the registered requests. RID shall be used to follow up the advertising result.
- Each server $S_{i=1:N}$ inserts the advertising request, the product advertisement, and the request number RID in its local database (step 6).

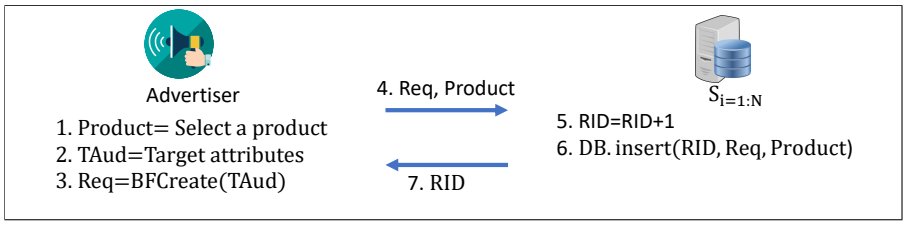

Fig. 7. Advertiser registration protocol (AdReg). The advertiser submits the request to all the servers and gets an identical RID from all of them (we assume servers have synchronized states).

Advertising Servers run this protocol for every unmatched pair of advertising request and a group of profiles, and store the matching result in their databases. Let GID and Req denote the group and the request that are to be matched, respectively. Figure 8 exhibits the overall interaction of servers for the advertising protocol. First, each server retrieves the profiles of the intended group, i.e., $Pf_1, ..., Pf_k$ and the request from its database (steps 1-2 of Figure 8). Next, servers proceed with three main phases 1) Aggregation, 2) Distributed Decryption, and 3) De-aggregation/Matching. While phases 1 and 3 are non-interactive, phase 2 requires servers interaction to run distributed decryption protocol. We stress that **both users and advertisers are offline during the matching procedure**, which is one of our main contributions.

1. *Aggregation:* Aggregation (step 4 of Figure 8) is run by each server locally. Algorithm 2 summarizes the entire procedure. Let R be the indices of the set bits in the advertising request as given in Equation 13.

$$R = \{i | r_i \in Req \text{ and } r_i == 1\} \tag{13}$$

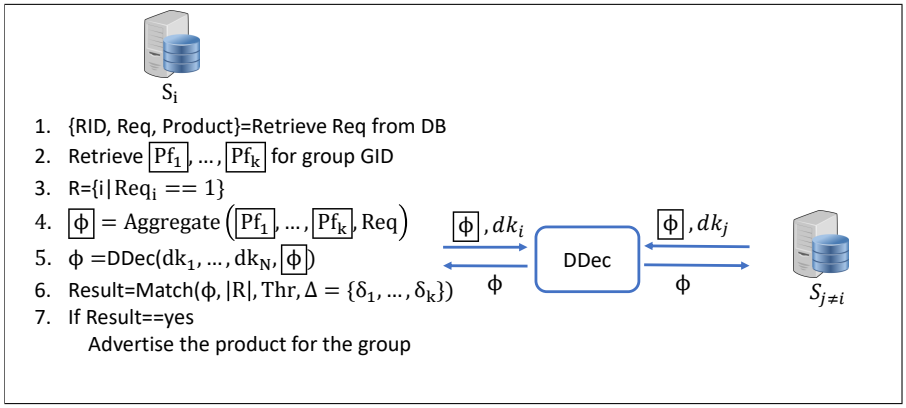

Fig. 8. Advertisement protocol (Ad) to match an advertising request (Req) to a group (GID). S_i communicates with all the other servers at step 5 to execute distributed decryption DDec protocol. All the servers run an identical set of instructions as indicated for S_i.

For each profile $Pf_{j=1:k}$ in the group, each server extracts the profile's elements in accordance to the index set R and then homomorphically sums them up (step 3 of Algorithm 2). This is indicated in Equation 14.

$$\boxed{\phi_j} = \prod_{\substack{i=1:p \\ i\in R}} \boxed{Pf_{j_i}} = \boxed{\sum_{\substack{i=1:p \\ i\in R}} b_{j_i} \cdot \delta_{\pi(j)}} = \boxed{\delta_{\pi(j)} \cdot \sum_{\substack{i=1:p \\ i\in R}} b_{j_i}} \tag{14}$$

Pf_{j_i} represents the i^{th} element of the j^{th} profile where $i \in [1, p]$ and $j \in [1, k]$. Observe that the final summation i.e., $\boxed{\phi_j}$ embodies the multiplication of the user's membership ID i.e., $\delta_{\pi(j)}$ with $\sum_{i\in R} b_{j_i}$ that is the sum of bit values of the profile in accordance to the set R. Let α_j denote $\sum_{i\in R} b_{j_i}$ (Equation 15).

$$\alpha_j = \sum_{\substack{i=1:p \\ i\in R}} b_{j_i} \tag{15}$$

Thus, $\boxed{\phi_j}$ would be

$$\boxed{\phi_j} = \boxed{\delta_{\pi(j)} \cdot \alpha_j} \tag{16}$$

After the computation of $< \boxed{\phi_1}, ..., \boxed{\phi_k} >$, each server aggregates them as given in Equation 17 (line 5 of Algorithm 2). Let $\boxed{\phi}$ be the final aggregate. Since all the servers hold a database with the same set of encrypted profiles, their local computation should all result in the same ciphertext $\boxed{\phi}$.

Otherwise, an equivocation is detected.

$$\boxed{\phi} = \prod_{j=1}^{k} \boxed{\phi_j} = \boxed{\sum_{j=1}^{k} \delta_{\pi(j)} * \alpha_j} \tag{17}$$

Algorithm 2 Aggregate($\boxed{Pf_1}, ..., \boxed{Pf_k}, Req$)

1: $R = \{i | r_i \in Req \text{ and } r_i == 1\}$
2: **for** $j = 1 : k$ **do**
3: $\quad \boxed{\phi_j} = \prod_{\substack{i=1:p \\ i \in R}} \boxed{Pf_{j_i}}$
4: **end for**
5: $\boxed{\phi} = \prod_{j=1}^{k} \boxed{\phi_j}$
6: **return** $\boxed{\phi}$

2. *Distributed Decryption:* All the servers use their decryption key shares to jointly decrypt $\boxed{\phi}$ through $DDec$ protocol (step 5 of Figure 8). We use ϕ to be the decrypted result as shown in Equation 18.

$$DDec(\phi) = \phi = \sum_{j=1}^{k} \delta_{\pi(j)} * \alpha_j \tag{18}$$

Recall that we employ a $DDec$ algorithm which stands malicious parties. As such, each server accompanies its computation result with a zero-knowledge proof of correct decryption [18]. Thus, if a corrupted server attempts decrypting a ciphertext different from $\boxed{\phi}$, or using a fake decryption key share then its proof will not be validated by honest servers and the cheating server is caught.

3. *De-aggregation/Matching:* Matching procedure is a non-interactive protocol that each server runs individually (step 6 of Figure 8). Algorithm 3 illustrates this procedure where the aim is to identify the total number of targeted profiles from Φ. It starts by dividing ϕ with the first largest membership ID namely, δ_k (line 3). The quotient of this division is equal to $\alpha_{\pi^{-1}(k)}$ where $\pi^{-1}(k)$ indicates the index of a group member with the membership ID δ_k. To realize why this is the case, let reformulate ϕ by extracting out the term of δ_k (to be the largest δ) as in Equation 19.

$$\frac{\phi}{\delta_k} = \frac{\delta_k \cdot \alpha_{\pi^{-1}(k)} + \sum_{m=1}^{k-1} \delta_m \cdot \alpha_{\pi^{-1}(m)}}{\delta_k} \tag{19}$$

Having known that $\alpha_{j=1:k}$ values are bounded by p together with Equation 12, we derive the following inequalities:

$$\sum_{m=1}^{k-1} \delta_m \cdot \alpha_{\pi^{-1}(m)} < \sum_{m=1}^{k-1} \delta_m \cdot p < \delta_k \tag{20}$$

Relying on Equation 20, the quotient of division in Equation 19 would be $\alpha_{\pi^{-1}(k)}$ and the remainder is $\sum_{m=1}^{k-1} \delta_m \cdot \alpha_{\pi^{-1}(m)}$.
If the quotient of division, i.e., $\alpha_{\pi^{-1}(k)}$ equals to $|R|$ (line 3) then one match is found in the group (line 4). This is true as explained next. First note that $\alpha_{\pi^{-1}(k)}$ contains the sum of bit values of the Bloom filter of the $\pi^{-1}(k)^{th}$ group member in accordance to the index set R (also see Equation 15). Hence, equality of $|R|$ and $\alpha_{\pi^{-1}(k)}$ indicates that the number of the set bits in the request (i.e., $|R|$) and the sum of the corresponding bits in a profile i.e., $\alpha_{\pi^{-1}(k)}$ are equal. This happens only when a profile contains all the attributes in the advertising request (i.e., every element of the profile corresponding to the non-zero elements of the request is also non-zero).
The procedure continues by setting the value of ϕ to the remainder of the division (line 6) and using the next largest δ as the divisor (line 2). At each step, the matching of one group member is identified and augmented to the total matches (line 4). If the number of matched users hits the threshold (line 8), then the advertisement is served for the entire group, otherwise, the group is skipped (lines 9-11).

Algorithm 3 $Match(\phi, |R|, Thr, \Delta = \{\delta_1, ..., \delta_k\})$

```
1: count = 0
2: for δ_j ∈ Δ, j = k : 1 do
3:    if Φ/δ_j == |R| then
4:       count = count + 1
5:    end if
6:    φ = φ  mod  δ_j
7: end for
8: if Thr ≤ count then return Yes
9: else
10:   return No
11: end if
```

Profile update: The profile update corresponds to replacing the existing profile with a new one. The procedure is identical to the profile registration except that servers do not redistribute membership identifiers and the updater uses his prior identifier and remains in the same group. In fact, the user interacts with servers by following only the steps 5-9 of Figure 6.

Performing profile update in a group-based advertising approach comes with the security concern [8], where servers can analyze the group matching results

Overhead\Entity	User	Advertiser	$S_1, ...S_N$
User Registration	$O(p \cdot \lambda)$	-	$O(p \cdot \lambda)$
Advertisement	-	-	$O(k \cdot \|R\| + N \cdot \lambda)$

(a) Privado Asymptotic Performance

Overhead\Entity	User	Advertiser	Provider	PSP
User Registration	$O(p)$	-	-	-
Advertisement	-	-	$O(k \cdot \|R\|)$	$O(k \cdot \|R\|)$

(b) PPAD [8] Asymptotic Performance

Table 1. Computation Complexity based on the group multiplications. k: number of users per group. $|R|$: number of set bits in the advertising request. p: size of the Bloom filter. N: the total number of servers. Advertiser registration does not involve any cryptographic operation hence is not included in the tables.

before and after the update operation and hence identify which attributes are modified in the updated profile. As an example, assume that an advertising request targeting an attribute x had no target customer in a particular group. Later on, a member of that group updates his profile. After that update, the number of matched profiles (for attribute x) increases to one. Thus, it is revealed that the updated profile contains attribute x.

To address this issue, we propose batch update, i.e., profile updates are applied when all the group members (of a single group) hand over at least one update. Assume the j^{th} member of the group wants to update his profile. He runs $PCreate$ algorithm (algorithm 1) over a new set of attributes. Let Op_j be the output tuple. User submits Op_j to all the servers. Servers copy this update in their databases but would not replace the old profile with this new one (thus will not consider it for any advertising). Instead, servers wait until there would be an update for every member of that group. Till then, the user may attempt multiple updates and servers would only consider the latest one. Once $Op_1, ..., Op_k$ are received from all the k members of the group, servers replace the all old profiles in that group with the updated ones. Henceforth, advertising is run over the new set of profiles. Notice that from now on any changes that happen to the group matching result (after the batch update) cannot be linked to a particular group member since all of the group members updated their profiles (and by the minimal assumption, at least two updates were performed by the honest members).

5 Complexity and Performance

5.1 Complexity

We demonstrate the computational complexity based on the number of group multiplications. The complexity analysis are illustrated in Table 1.a.

User: User performs $O(p)$ encryption operations to create $\boxed{BF}$ and $\boxed{Pf}$ as part of his profile. Each encryption requires $O(\lambda)$ group multiplications. Further, for each element of $\boxed{BF}$ and $\boxed{Pf}$, the user generates proof of plaintext range and proof of correct multiplication. Each proof is of $O(\lambda)$. Thus, the overall running time complexity of user is $O(p \cdot \lambda)$.
Advertiser: An advertiser only creates a plaintext Bloom filter, hence carries no cryptographic operations.
Servers: We analyze the running time complexity of servers for each protocol separately as follows.

- **User registration**: During this protocol and for each group of profiles, servers run mix-net to create a fresh permutation of the membership identifier set. We use the verifiable shuffle scheme proposed by [38]. As such, in each instance of mix-net, each server carries $O(k \cdot \lambda)$ operations (or equivalently $O(\frac{k \cdot \lambda}{k}) = O(\lambda)$ operations per group member).
  Additionally, servers receive the encrypted profiles whose correctness must be checked. That is, servers verify σ^{POPR} and σ^{POCM} that are generated for p distinct elements of each profile. Verification of each proof incurs constant overhead in λ. Thus, in total, profile verification costs $O(p \cdot \lambda)$ at each server. In total, $O(p \cdot \lambda)$ is the overhead of profile registration on each server.
- **Advertising**: In this protocol, servers locally compute the $\boxed{\Phi}$ value then decrypt it. The computation of $\boxed{\Phi}$ requires $(k \cdot |R|) + k - 1$ operations. Then, each server uses its own share of the decryption key to decrypt $\boxed{\Phi}$ which is done in $O(\lambda)$. Additionally, each server must verify the computation (decryption) of $N - 1$ other servers which results in $O((N-1) \cdot \lambda)$ more operations. In total, $O(k \cdot |R| + N \cdot \lambda)$ is a load of advertising on each server. We emphasize that N adds only an additive overhead to the advertising run-time.

5.2 Concrete Performance

We investigate the performance of Privado by simulating the advertising protocol on a computer with an Intel Xeon 2.93 GHz CPU, 80GB RAM, and Ubuntu 16.4 operating system. We deploy Paillier encryption scheme with 2048 bit modulus. Our performance results are taken using Fiat-Shamir heuristic implementation.

We generate 1000 random profiles with 400 attributes (based on our personal experience from Facebook advertising as well as due to [8], 400 attributes are approximately the maximum number) as well as 100 advertising requests with 30 random attributes (for randomly generated profiles, almost no match is found for an advertisement with more than 30 attributes [8]). A Bloom filter accommodating 400 attributes has a size of $p = 6848$. An advertising request with 30 attributes contains 294 set bits in its Bloom filter representation (i.e., $|R| = 294$). We group the profiles using group sizes 2-20, create their encrypted format and attempt to run advertising protocol over the resultant groups. The results are shown in Figure 9.

The results assert that the server's overhead linearly scales with the group size and the number of servers. In particular, the group size impacts the server's running time to aggregate a group of profiles, i.e., the computation of $\boxed{\Phi}$, whereas the number of servers influences the running time of decryption of $\boxed{\Phi}$ (during which each server should verify the decryption integrity of $N-1$ other servers). Adding a new server to the system will increase the run time of each server by $30ms$ which is the time required to verify the decryption of one server.

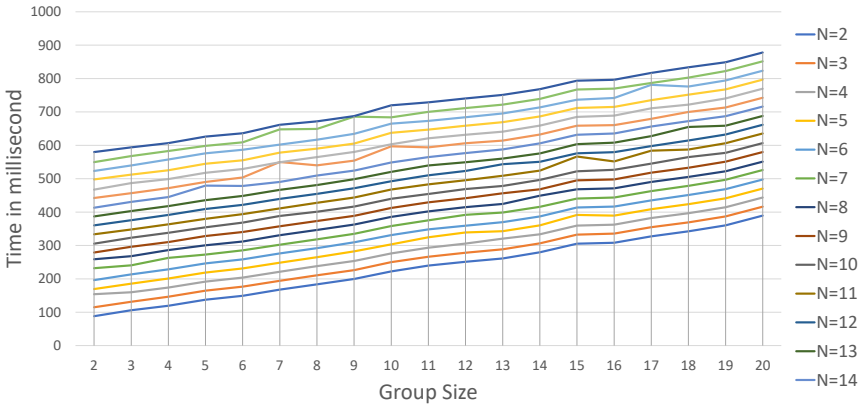

Fig. 9. Running time of servers in Advertising protocol over the different number of servers and group sizes. N indicates the number of servers.

Recommended Number of Servers: In Privado, we utilize multiple servers in order to provide better privacy for the users, i.e., the privacy holds unless an adversary obtains control of all the servers. Thus, increasing the number of servers would make the job of the adversary harder (hence results in a stronger privacy guarantee). In contrast, the number of servers negatively influences the advertising running time and degrades the performance. Thus, the selection of N is a trade-off between privacy and performance. Note that, due to the privacy concern, N cannot be less than 2. For any value of N greater than 2, the best candidate can be set according to the computational power of servers and the desired performance. For instance, in a system with group size 11, if the desired running time of advertising protocol is less than 400 milliseconds, then the maximum value of N (which is the best for user privacy) would be 8 (in Figure 9, the running time line of $N = 8$ is the closest line to the intersection of group size 11 with the running time 400 milliseconds).

Privado vs PPAD: We compare the advertising running time of Privado with PPAD [8]. For the results to be comparable, we deploy 2 servers for Privado similar to PPAD. The computation complexity of PPAD is given in Table 1.b.

With respect to the computation complexity, both servers in Privado and PPAD run an identical set of instructions to perform aggregation (computation of $\boxed{\Phi}$) and decryption. However, in Privado, servers have to verify each others' decryption results in order to provide security against malicious servers. In fact, the additional term $N \cdot \lambda$ in Table 1.a under the advertising row asserts this fact. That is, each server of Privado has to carry $O(\lambda)$ group multiplications to verify the decryption result of another server.

The simulation results (shown in Figure 10) comply with the asymptotic analysis. In order to clarify this observation, in Figure 10, we plot the run time of Privado under both honest-but-curious (HbC) and malicious adversarial models. In the HbC curve of Privado, we exclude the verification run time (i.e., servers do not verify the ZKP of correct decryption of one another) assuming that they all trustfully follow the protocol. As it is apparent from the graphs, PPAD and Privado run identically in the HbC model. The malicious setting of Privado imposes 30 ms to the total run time (that is the verification run time), which is the time that each server spends to verify other server's output. This difference causes PPAD to be 1.2 times faster than Privado in the two-server setting. For instance, under the group size 7, PPAD performs advertising in 140 ms whereas Privado runs in 168 ms ($\frac{168}{140} = 1.2$). But, we emphasize that this computational cost has enabled Privado to stand a stronger adversarial model and deliver more robust security guarantee for its users.

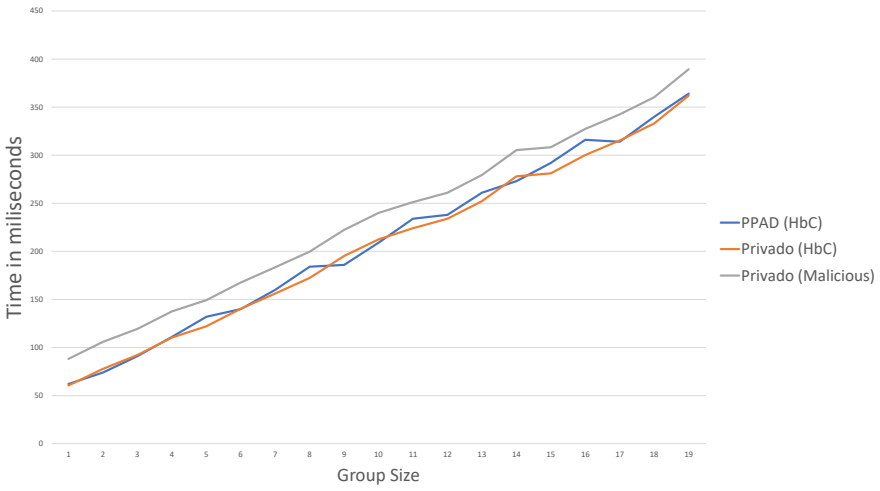

Fig. 10. The comparison of running time of advertising protocol in Privado and PPAD over different group sizes. Both systems deploy two servers. In the HbC model of Privado, servers do not verify each other's computations (assuming servers trustfully follow the protocol).

6 Security

We analyze the security of Privado against an active adversary who controls at most $N-1$ servers, $k-2$ users per group, and an arbitrary number of advertisers (as discussed earlier, more than $k-2$ colluding users per group cannot be made secure by any technique, and clearly, if all servers are adversarial then privacy is not applicable). We consider a malicious adversary which may refuse to follow the protocols' instructions and act in an arbitrary way. We present the security definition of user privacy in Section 6.1 (originally proposed by [8]) and provide a full security proof against this most powerful adversary in Section 6.2.

6.1 Security Definition

We utilize the following game to illustrate the formal definition of user privacy. The game is played between an adversary and a challenger. The adversary directs all the colluding malicious entities in the system i.e. $N-1$ servers, $k-2$ users per group, and any number of advertisers. On the other hand, the challenger controls the rest of parties which are honest, i.e., 2 of the group members, 1 server and some of the advertisers. In this game, the adversary selects two sets of attributes for the honest users of a group and asks the challenger to register the honest users using those sets. The challenger assigns the sets randomly to the honest users and registers them accordingly. The adversary's task is to guess which attribute set is registered under which username. In secure group-based advertising, the adversary would not be able to guess the correct assignment with better than a negligible advantage. Thus, even though the adversary has the power to know and even set the attributes of the two honest users, he cannot decide which user has which attribute set.

Note that, this security definition perfectly copes with the security challenges we discussed in Section 4.1. In particular, if the adversary manages to mount any of those attacks, then it would be able to break this game. For example, if the adversary manages to reuse one of the honest members profile for a corrupted member (mount replay attack), then (as we explained in Section 4.1) this would result in a specific pattern in the total number of target users of the group. As such, adversary can precisely learn the assignment of attributes to honest users and win the game. Therefore, the failure of the adversary in this game means the resilient of the design against all those attacks.
We consider $UPrivacy_{\mathcal{A}}(\lambda)$ a probabilistic experiment defined in terms of a game played between adversary $\mathcal{A}$ and a challenger, as given below [8].

User privacy experiment $UPrivacy_{\mathcal{A}}(\lambda)$**:**

1. Adversary and challenger are given the security parameter λ. They execute the Initialization protocol.
2. Adversary registers $k-2$ users to the group with the GID^* to be attacked. Subsequently, the challenger takes the role of the other two users whose usernames are denoted by $Uname_0$ and $Uname_1$. This step can be executed after step 4 as well.
3. Adversary outputs two attribute sets Att_0 and Att_1 to be assigned to the two honest users in the group to be attacked.
4. Challenger selects a bit value b randomly. He assigns Att_b and $Att_{\hat{b}}$ to $Uname_0$ and $Uname_1$, respectively. Finally, the challenger creates two profiles accordingly and runs the UReg (User Registration) protocol on behalf of these two users together with the adversary.
5. challenger registers advertisers upon adversary's request with the attributes the adversary provides. This step can be run polynomially-many times.
6. The advertising protocol is executed jointly by the challenger and the adversary for polynomially many advertising requests.
7. The adversary outputs a bit $b^{'}$. If $b == b^{'}$ then the output of the experiment is 1 indicating that the adversary succeeds, otherwise it is 0.

Definition 1. *Privado protects user privacy against an active adversary if for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\lambda)$ *where* λ *is the security parameter such that:*

$$Pr[UPrivacy_{\mathcal{A}}(\lambda) = 1] \leq \frac{1}{2} + negl(\lambda) \tag{21}$$

6.2 Formal Security Proof

Proof Overview: Our proof consists of two parts. In the first part, illustrated in Theorem 1, we show that breaking user privacy of Privado would compromise the IND-CPA security of the underlying encryption scheme $TEnc$. This is done through the construction of a simulator B which ties user privacy game to the IND-CPA game. As the second part, we supply Theorem 2 in which we prove that B runs indistinguishable from the real challenger due to the IND-CPA security of the deployed encryption $TEnc$.

Note that in Privado, for efficiency, we deploy Fiat-Shamir heuristic to enable non-interactive ZKPs for POPR, POCM and the proof system of the verifiable shuffle scheme. Thus, the security of F^R_{POPR}, F^R_{POCM}, and F^R_{VS} are in the random oracle model (and subsequently, the proof of Theorem 1). However, we emphasize that our security proof also works intact in the standard model by utilizing interactive ZKPs.

Theorem 1. *Privado preserves user privacy as given in Definition 1, in F^R_{POPR}, F^R_{POCM}, F^R_{VS}, F_{THRES} hybrid model, assuming that the $TEnc$ encryption scheme is IND-CPA secure.*

Proof: If there exists a PPT adversary $\mathcal{A}$ who breaks user privacy with $\epsilon(\lambda)$ advantage, then we can construct a PPT adversary B who breaks the IND-CPA security of $TEnc$ with the same advantage. Let $h \in [1, N]$ be the index of the honest server that is run by B whereas $\bar{h}$ be the set of $N-1$ servers controlled by $\mathcal{A}$. B also simulates the ideal functionalities F^R_{POPR}, F^R_{POCM}, F^R_{VS}, and F_{THRES}. B works as follows.

1. Simulator B engages with $\mathcal{A}$ to execute the initialization phase. B, receives the encryption key pk and the security parameter 1^λ from IND-CPA challenger. B, simulating F_{THRES}, waits to obtain $(S_i, Generate, 1^\lambda)$ message from each server $S_i \in \bar{h}$ and then asks $\mathcal{A}$ for a randomness. Upon the receipt of randomness, B hands over pk (obtained from IND-CPA challenger) to $\mathcal{A}$. If $\mathcal{A}$ responds with *continue*, B proceeds with the rest of executions, otherwise sends *abort* to $P_{i \in h \cup \bar{h}}$ and terminates.

   Next, one of the servers namely S_j creates the membership identifiers and their encryption. If $S_j \in \bar{h}$ (controlled by A) then two situations may happen. $\mathcal{A}$ either outputs the membership identifier set $\Delta = \{\delta_i\}_{i=1:k}$, its encryption $\boxed{\Delta} = \{\boxed{\delta_i} = Enc(\delta_i, r_i)\}_{i=1:k}$, together with the randomnesses $\{r_i\}_{i=1:k}$ or A outputs an empty set. In the former case, B verifies whether Δ satisfies Equation 12 and also whether $\boxed{\Delta}$ is the correct encryption of Δ using the given randomnesses. If $\mathcal{A}$ outputs an empty set then B generates Δ and $\boxed{\Delta}$ by itself and communicates Δ and $\boxed{\Delta}$ and the encryption randomnesses to $S_{i \in \bar{h}}$ to be verified by A.
2. (a) B and $\mathcal{A}$ run the mix-net. We assume shuffling starts from S_1 and ends with S_N. The initial input to the mix-net is $\boxed{\Delta_{\pi_0}} = \boxed{\Delta}$. For every $S_{i \in \bar{h}}$, $\mathcal{A}$ creates $\boxed{\Delta_{\pi_i}}$ by shuffling and re-encrypting $\boxed{\Delta_{\pi_{i-1}}}$ (π_{i-1} and π_i are the permutations of S_{i-1} and S_i, respectively). $\mathcal{A}$ calls F^R_{VS} to prove the correctness of its shuffle to every other servers including S_h. As such, $\mathcal{A}$ sends $\boxed{\Delta_{\pi_{i-1}}}$, $\boxed{\Delta_{\pi_i}}$, and the witness ω_i as the proof of correct shuffle to F_{VS-R}. A part of ω_i includes the permutation π_i (to indicate that $\boxed{\Delta_{\pi_i}}$ is the correct shuffle of $\boxed{\Delta_{\pi_{i-1}}}$). This means that B while simulating F^R_{VS} will learn this permutation. However, we remark that B will not use this information. B simulates F^R_{VS} and verifies the proof correctness. When the honest server S_h takes the turn, B simulates on behalf of it as follows. B populates $\boxed{\Delta_{\pi_h}}$ with k encryptions of a junk value δ' i.e., $\boxed{\Delta_{\pi_h}} = \{\boxed{\delta'}, ..., \boxed{\delta'}\}$. B delivers $\boxed{\Delta_{\pi_h}}$ to S_{h+1} (that is controlled by $\mathcal{A}$). B sends $(proof, S_h, S_i, (pk, \boxed{\Delta_{\pi_{h-1}}}, \boxed{\Delta_{\pi_h}}))$ to each $S_{i_{\in \bar{h}}}$ playing as

F^R_{VS}. If $h = N$ then the honest server's (B's) output $\boxed{\Delta_{\pi_h}}$ constitutes the final output of the mix-net.

(b) $\mathcal{A}$ creates $k-2$ profiles on behalf of corrupted members. As such, $\mathcal{A}$ communicates $\boxed{BF_j}$, and $\boxed{Pf_j}$ with S_h (j indicates the index of corrupted member). Also, $\mathcal{A}$ proves the correctness of profiles to S_h by invoking F^R_{POPR} and F^R_{POCM} for every element of $\boxed{BF_j}$ and $\boxed{Pf_j}$, respectively. B shall act as F^R_{POPR} and F^R_{POCM} and verify the correctness of profiles. Note that according to Equation 9, B while simulating F^R_{POPR}, learns the profiles of corrupted members. That is, for each encrypted element $\boxed{b_{j,i=1:p}}$ of $\boxed{BF_j}$ (for each corrupted member j controlled by $\mathcal{A}$), $\mathcal{A}$ submits a witness ω to B which contains the corresponding plaintext $b_{j,i}$ (together with the encryption randomness). This enables B to learn all the bit value of the Bloom filter BF_j.

3. $\mathcal{A}$ outputs two attribute sets Att_0, Att_1 to be used for the honest users.
4. B generates two Bloom filters out of Att_0 and Att_1 namely BF_0 and BF_1. B sends $M_0 = \{BF_0, BF_1\} = \{b_{0,1}, ..., b_{0,p}, b_{1,1}, ..., b_{1,p}\}$ and $M_1 = \{BF_1, BF_0\} = \{b_{1,1}, ..., b_{1,p}, b_{0,1}, ..., b_{0,p}\}$ to the IND-CPA challenger and obtains $C = \{\boxed{b_{b,1}}, ..., \boxed{b_{b,p}}, \boxed{b_{\hat{b},1}}, ..., \boxed{b_{\hat{b},p}}\} = \{\boxed{BF_b}, \boxed{BF_{\hat{b}}}\}$. B homomorphically multiplies δ' into each encrypted Bloom filter $\boxed{BF_b}$ and $\boxed{BF_{\hat{b}}}$ and constructs $\boxed{Pf_b}$ and $\boxed{Pf_{\hat{b}}}$, respectively. B registers $\boxed{BF_b}$, $\boxed{Pf_b}$ under $UName_0$ and $\boxed{BF_{\hat{b}}}$, $\boxed{Pf_{\hat{b}}}$ for $UName_1$. B acts as F^R_{POPR} and F^R_{POCM} to prove the correctness of profiles to each server $S_{i\in\bar{h}}$. That is, for the j^{th} element of BF_b i.e., $\boxed{b_{0,j}}$ where $j \in [1, p]$, B sends (proof, $UName_0$, S_i, (pk, $\boxed{BF_{b,j}}$, $[0, 1]$)) to each $S_{i\in\bar{h}}$. Also, for the j^{th} element of $\boxed{Pf_b}$ B sends (proof, $UName_0$, S_i, (pk, $\boxed{BF_{b,j}}$, C_b, $\boxed{Pf_{b,j}}$)) to each $S_{i\in\bar{h}}$. B acts similarly for $UName_1$, $\boxed{BF_{\hat{b}}}$ and $\boxed{Pf_{\hat{b}}}$
5. $\mathcal{A}$ and B start registering arbitrary advertising requests to the system.
6. $\mathcal{A}$ invokes the advertising protocol for an arbitrary advertising request Req. This step can be repeated polynomially many times. B computes the aggregation as $\boxed{\Phi}$ based on the registered profile. Next, B, simulating F_{THRES}, waits for all the other servers $S_{i\in\bar{h}}$ to request decryption of $\boxed{\Phi}$ to compute the decryption result. Observe that B does not own the decryption power so cannot execute decryption. However, in step 2.b, B acting as F^R_{POPR} could learn the content of corrupted users' profiles. Also, B knows the content of honest users profiles due to step 3. Thus, B can craft the decryption result i.e., Φ on its own. As such, B first computes the matching results of corrupted members i.e., α_3,...α_k using the Bloom filters extracted in step 2 (α_is is defined in Equation 15). Then, B calculates the individual matching results of the honest members i.e., α_1 and α_2 based on BF_0 and BF_1, respectively. B needs to associate each matching result with a membership

identifier (as given in Equation 17). Therefore, B shuffles Δ under a random permutation π and associates j^{th} element of Δ_π i.e., $\delta_{\pi(j)}$ with the j^{th} group member. B performs shuffling only once and then keeps using the same assignment for the rest of advertising requests. B outputs the aggregate value Φ as $\sum_{j=1:k} \delta_{\pi(j)} \cdot \alpha_j$.

Note that B acting as F_{THRES} is supposed to perform decryption of a ciphertext (i.e., encrypted aggregation) upon the request of all the N servers. Thus, B would only attempt decryption of ciphertexts that are agree with its own (i.e., the honest server's) local computations for which B knows the corresponding plaintext. Thus, B is always able to make a correct decryption indistinguishable from F_{THRES}. If $\mathcal{A}$ sends a decryption request for an arbitrary ciphertext which does not correspond to any aggregation computed by B, then B would never decrypt it.

7. $\mathcal{A}$ outputs a bit value b'. B delivers the same value to the IND-CPA challenger.

B carries polynomial operations at each step hence runs in polynomial time. Also, B simulates user privacy game to $\mathcal{A}$ indistinguishable from the original game as discussed next. Steps 1 is done as instructed in the real protocol. In step 2, B replaces all the membership IDs i.e., $\delta_1, ..., \delta_k$ with δ' which remains unnoticed to $\mathcal{A}$ due to the IND-CPA security of the encryption scheme (this is formally proven in in Theorem 2). Step 3 is run as expected. In step 4, B creates the content of the two honest users flawlessly. Though, B does not know the real content of $\boxed{BF_b}$ and $\boxed{BF_{\hat{b}}}$ to generate the profile correctness proof, B itself simulates F^R_{POPR} and F^R_{POCM} which enable him to approve to the adversary $\mathcal{A}$ that the profiles of honest members are constructed appropriately. Step 5 is executed as expected. In step 6, the decrypted aggregate value Φ constructed by B is certainly admissible to $\mathcal{A}$ since the decryption is modeled by the ideal functionality F_{THRES}.

Notice that the execution of the protocol at steps 1-3 and 5 convey no useful information regarding the attribute assignment of the honest members. This is the case since the simulator B uses identical membership identifiers δ' for all the group members including the honest ones. Thus, the only way to learn which attribute is assigned to which honest member is through the content of registered profiles. This should not be possible as the utilized encryption scheme is IND-CPA secure i.e., the profiles reveal no information about the underlying attributes. Hence, if $\mathcal{A}$ guesses bit b' with non-negligible advantage, the IND-CPA security of the underlying encryption is broken. In particular, the output bit b' asserts that $Att_{b'}$ is assigned to $UName_0$ i.e., $BF_{b'}$ is used in $UName_0$ registration. Recall that the profile of $UName_0$ is constructed based on the first part of IND-CPA challenger's output. Therefore, the IND-CPA challenger's output must be the encryption of $M_{b'}$ i.e., $\{\boxed{BF_{b'}}, \boxed{BF_{\hat{b'}}}\}$. Assuming that $\mathcal{A}$ has non-negligible advantage $\epsilon(\lambda)$ in winning the user privacy game, then B by outputting the same b' also breaks the IND-CPA game with non-negligible

advantage $\epsilon(\lambda)$. This is a contradiction with the initial assumption stated in Theorem 1. Thus, $\epsilon(\lambda)$ is negligible. ■

As for the indistinguishability of $\{\delta', ..., \delta'\}$ from real membership IDs i.e., $\{\delta_1, ..., \delta_k\}$ in the mix-net execution, we construct a modified simulator B' which runs identical to B except that at step 3.a, during the mix-net execution, it uses the real membership identifiers. In particular, B' executes step 2.(a) honestly (not with the junk identifiers).

Next we prove that $\mathcal{A}$ cannot distinguish its interaction with B and B' unless the underlying encryption scheme is not IND-CPA secure. Let $UPrivacy_{A,B'}(\lambda)$ indicate the user privacy experiment run between $\mathcal{A}$ and B'.

Theorem 2. *In F^R_{POPR}, F^R_{POCM}, F^R_{VS}, F_{THRES} hybrid model and assuming that the $TEnc$ encryption scheme is IND-CPA secure then*

$$|Pr[UPrivacy_{\mathcal{A},B}(\lambda) = 1] - Pr[UPrivacy_{\mathcal{A},B'}(\lambda) = 1]| < negl(\lambda) \tag{22}$$

Proof: If $\mathcal{A}$ can distinguish between its interaction with B and B' with non-negligible advantage ϵ', i.e.,

$$|Pr[UPrivacy_{\mathcal{A},B'}(\lambda) = 1] - Pr[UPrivacy_{\mathcal{A},B}(\lambda) = 1]| = \epsilon' \tag{23}$$

then we can construct an adversary D who can break the IND-CPA security of the encryption scheme with ϵ' advantage. D interacts with $\mathcal{A}$ as below.

1. D runs identical to B.
2. (a) D and $\mathcal{A}$ run the mix-net. Shuffling starts from S_1 and ends with S_N. For every $S_{i \in \bar{h}}$, $\mathcal{A}$ sends $\boxed{\Delta_{\pi_{i-1}}}$ and $\boxed{\Delta_{\pi_i}}$ (π_{i-1} and π_i are the permutations of S_{i-1} and S_i, respectively) together with a witness ω_i as the proof of correct shuffle to F_{VS-R}. D simulates F^R_{VS} and verifies the proof. When S_h takes the turn, D simulates on behalf of it as follows. D selects a random permutation π_h and permutes the plaintext Δ as $\Delta_{\pi_h} = \{\delta_{\pi(1)}, ..., \delta_{\pi(k)}\}$. D constructs $M_0 = \{\delta_{\pi(1)}, ..., \delta_{\pi(k)}\}$ and $M_1 = \{\delta', ..., \delta'\}$ and hands over to the IND-CPA challenger. δ' is a junk identifier. As the result, D receives ciphertext $C = \{C_1, ..., C_k\}$. He sends $\boxed{\Delta_{\pi_h}} = \{C_1, ..., C_k\}$ to S_{h+1} (that is controlled by $\mathcal{A}$). D simulates F^R_{VS} and approves the correctness of $\boxed{\Delta_{\pi_h}}$ to $\mathcal{A}$.
   (b) $\mathcal{A}$ registers $k-2$ users into the system. D acts identical to B. Recall that in this step, B will learn the content of registered profiles i.e., $BF_3, ..., BF_k$.
3. $\mathcal{A}$ outputs two attribute sets Att_0 and Att_1.
4. D generates two Bloom filters out of Att_0 and Att_1 namely $BF_0 = \{b_{0,1}, ..., b_{0,p}\}$ and $BF_1 = \{b_{1,1}, ..., b_{1,p}\}$. He constructs $\boxed{Pf_0}$ using BF_0 and the membership ID C_1, and $\boxed{Pf_1}$ using BF_1 and the membership ID C_2. Next, D flips a coin $b \leftarrow \{0, 1\}$ and registers $\boxed{BF_b}$, $\boxed{Pf_b}$ under $UName_0$ and $\boxed{BF_{\hat{b}}}$, $\boxed{Pf_{\hat{b}}}$ for $UName_1$. Then, D acts as F^R_{POPR} and F^R_{POCM} to prove the

correctness of profiles to each server $S_{i\in\bar{h}}$. That is, for the j^{th} element of BF_b i.e., $\boxed{b_{0,j}}$ where $j \in [1,p]$, D sends (proof, $UName_0$, S_i, (pk, $\boxed{BF_{b,j}}$, $[0,1]$)) to each $S_{i\in\bar{h}}$. Also, for the j^{th} element of $\boxed{Pf_0}$ D sends (proof, $UName_0$, S_i, (pk, $\boxed{BF_{b,j}}$, C_b, $\boxed{Pf_{b,j}}$)) to each $S_{i\in\bar{h}}$. D acts similarily for $UName_1$, $\boxed{BF_{\hat{b}}}$ and $\boxed{Pf_{\hat{b}}}$.
Observe that in the view of adversary $\mathcal{A}$, D runs this step the same as B (since similar to B, D also ends up with a random assignment of profiles to the honest users).

5. D and $\mathcal{A}$ register their own advertising requests.
6. $\mathcal{A}$ invokes the advertising protocol for an arbitrary advertising request Req. D acts the same as B.
7. $\mathcal{A}$ outputs a bit value b'. If $b == b'$ then D outputs 0, otherwise 1.

Clearly, D operates in polynomial time. D simulates indistinguishable from the real challenger (as well as indistinguishable from B) in user privacy game. All the steps are run identical except the shuffle operation. If the bit choice of IND-CPA challenger is 0 then D receives $C = \{\boxed{\delta_1}, ..., \boxed{\delta_k}\}$. In such case, the view of $\mathcal{A}$ is identical to its interaction with B'. If $C = \{\boxed{\delta'}, ..., \boxed{\delta'}\}$ then $\mathcal{A}$ experiences an interaction with B. According to Equation 23 we have:

$$\begin{aligned}\epsilon'(\lambda) =& |Pr[UPrivacy_{\mathcal{A},B'}(\lambda) = 1] - Pr[UPrivacy_{\mathcal{A},B}(\lambda) = 1]| \\ & |Pr[b == b' | C = \{\boxed{\delta_{\pi(1)}}, ..., \boxed{\delta_{\pi(k)}}\}] - Pr[b == b' | C = \{\boxed{\delta'}, ..., \boxed{\delta'}\}]| = \\ & |Pr[\text{D outputs } 0 | M = M_0] - Pr[\text{D outputs } 0 | M = M_1]| = \\ & |Pr[output(PubK^{CPA}_{D,TEnc}(\lambda, 0)) = 0] - Pr[output(PubK^{CPA}_{D,TEnc}(\lambda, 1)) = 0]|\end{aligned}$$

The last equality holds due to IND-CPA security of $TEnc$ (Equation 5). If $\epsilon'(\lambda)$ is non-negligible, then D can break the IND-CPA security of $TEnc$ with the non-negligible advantage. This yields to a contradiction with Theorem 2. Thus, $\epsilon'(\lambda)$ is negligible which implies that $\mathcal{A}$ cannot distinguish its interaction with B and B'. ■

7 Related Works

In this section, we describe the related works and compare them with Privado. We first discuss PPAD [8] as our main counterpart. Then, we additionally investigate four other different areas that are the most similar to the present context and identify their shortcomings when applied to the OSN advertising scenario.

PPAD: Privacy-Preserving Group-Based Advertising in Online Social Network: PPAD is a secure advertising system which deploys group-based matching notion and makes use of two servers called the OSN server and the Privacy Service Provider (PSP). PPAD satisfies advertising transparency as well as provable user privacy. However, it has two main shortcomings. Firstly, in PPAD,

the data decryption power is given to the PSP, and hence user privacy is tied to the fact that the OSN provider would not be able to corrupt that single PSP. This is a huge trust assumption to be placed on a single party and might be very hard to achieve in real life. Secondly, the adversary is an honest but curious entity which is supposed to precisely follow the protocol descriptions.

Secure Online Behavioral Advertising Systems (SOBA): In SOBA, a server named broker is connected to a set of web page owners. As users visit different web pages, the web page owners communicate this data with the broker. The broker then is able to create a behavioral profile for each user according to the visited web pages. The advertising companies pay to the broker to gain users profiles and be able to advertise for their target users on the web pages. The proposals in the context of SOBA are not applicable to the OSN advertising due to the lack of advertising transparency [4, 23, 46] and provable security [16].

Server Aided Private Set Intersection (S-PSI): In an S-PSI protocol, two parties, holding their own private sets of elements, are willing to compute their sets' intersection with the help of a third server. In this context, solutions either violate the advertising transparency [30, 29, 25, 37] or when transplanted in the OSN advertising they fall short in providing user privacy (as we define in advertising context) [29, 48]. In fact, PSI protocols usually make two assumptions that conflict with our required features: 1- they assume that the data holders can directly communicate some secret information unbeknown to the server which is against advertising transparency 2- they assume that set holders and the server are mutually untrusted and non-colluding [36, 40] whereas in our setting advertiser (one of the set holders) is cooperating with the server. Moreover, none of the prior PSI protocols offer a group-based method (matching multiple sets with one) and such an extension is not trivial.

Public Key Encryption with Keyword search (PEKS): PEKS is an attempt to outsource searching over data that is encrypted using a public key system. Encrypted data is usually outsourced to an external server who is responsible to respond to the data owner's search queries. The first issue with PEKS solutions is that they achieve privacy by assuming that the querier and the server are not colluding. This is in contradiction to our assumption (advertiser and server collude). Server and querier collusion enables keyword guess attack [19, 21, 49, 7, 32] which is against user privacy in the advertising scenario.

Server Aided Two/Multi-Party Computation (S-2PC/MPC): In S-2PC/MPC protocols two/multiple parties holding their own private data wish to compute a function of their inputs by the help of server(s). Parties only learn the output of function and nothing else. The main issue in S-2PC protocols is that the two parties are required to be online and involved per function evaluation to communicate some secret information [26, 28, 20, 33, 24, 9, 5]. This is in contrast to the advertising transparency. Moreover, the data holders are not supposed to collude with the server [47, 10] which is not the case in our setting. Indeed, the collusion of servers with any of the parties would violate our defined user privacy.

8 Conclusion

In the secure proposals of online social networks, users encrypt their data before sharing with OSNs. Data encryption blocks accessibility to the plaintext profiles, disabling the advertising service. To address this problem, we proposed Privado as a privacy-preserving group-based advertising system for secure OSNs. Our design is comprised of N servers each provided by an independent authority. We utilized the group-based advertising notion to enable user privacy, i.e., to hide the identity of the exact target customers. As such, users are divided into groups of size k at the registration time and then submit their profiles in an encrypted format. Advertisers submit their requests as plaintext. Servers find the target groups for the advertising requests. User privacy (i.e., the unlinkability of group matching result to the individual group members) is preserved against a malicious adversary who corrupts $N-1$ servers, $k-2$ members of each group and any number of advertisers. We formally defined and proved user privacy. Our advertising scheme enjoys advertising transparency where the entire process of matching groups to the advertising requests are done independent of users' and advertisers' collaboration. We performed experimental simulations and measured the advertising running time over a various number of servers and group sizes. We additionally discussed the optimum value of N, the number of servers, with regard to user privacy and advertising running time.

Acknowledgements

We acknowledge the support of the Turkish Academy of Sciences, the Royal Society of UK Newton Advanced Fellowship NA140464, TÜBİTAK (the Scientific and Technological Research Council of Turkey) under the project number 115E766, and EU COST Action IC1306.

References

1. P. Ananth, N. Chandran, V. Goyal, B. Kanukurthi, and R. Ostrovsky. Achieving privacy in verifiable computation with multiple servers–without fhe and without pre-processing. In *International Workshop on Public Key Cryptography*, pages 149–166. Springer, 2014.
2. R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: an online social network with user-defined privacy. In *ACM SIGCOMM*, 2009.
3. A. Barenghi, M. Beretta, A. Di Federico, and G. Pelosi. Snake: An end-to-end encrypted online social network. In *ICESS*. IEEE, 2014.
4. D. Biswas, S. Haller, and F. Kerschbaum. Privacy-preserving outsourced profiling. In *CEC*. IEEE, 2010.
5. M. Blanton and F. Bayatbabolghani. Efficient server-aided secure two-party function evaluation with applications to genomic computation. *Proceedings on Privacy Enhancing Technologies*, 2016(4):144–164, 2016.
6. B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

7. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *International conference on the theory and applications of cryptographic techniques*, pages 506–522. Springer, 2004.
8. S. T. Boshrooyeh, A. Küpçü, and Ö. Özkasap. PPAD: Privacy Preserving Group-Based ADvertising in Online Social Networks. In *IFIP Networking 2018*, pages 541–549, Zurich, Switzerland, 2018. IEEE.
9. H. Carter, B. Mood, P. Traynor, and K. Butler. Outsourcing secure two-party computation as a black box. In *Cryptology and Network Security*. 2015.
10. H. Carter, B. Mood, P. Traynor, and K. Butler. Secure outsourced garbled circuit evaluation for mobile devices. *Journal of Computer Security*, 24(2):137–180, 2016.
11. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 280–300. Springer, 2001.
12. E. De Cristofaro, C. Soriente, G. Tsudik, and A. Williams. Hummingbird: Privacy at the time of twitter. In *Security and Privacy (SP)*. IEEE, 2012.
13. A. J. Feldman, A. Blankstein, M. J. Freedman, and E. W. Felten. Social networking with frientegrity: Privacy and integrity with an untrusted provider. In *USENIX*, 2012.
14. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology—CRYPTO'86*, pages 186–194. Springer, 1986.
15. O. Goldreich. *Foundations of cryptography: volume 1, basic tools*. Cambridge university press, 2007.
16. S. Guha, B. Cheng, and P. Francis. Privad: practical privacy in online advertising. In *NSDI*, 2011.
17. C. Hazay and Y. Lindell. *Efficient secure two-party protocols: Techniques and constructions*. Springer Science & Business Media, 2010.
18. C. Hazay, G. L. Mikkelsen, T. Rabin, and T. Toft. Efficient rsa key generation and threshold paillier in the two-party setting. In *Cryptographers' Track at the RSA Conference*, pages 313–331. Springer, 2012.
19. D. He, M. Ma, S. Zeadally, N. Kumar, and K. Liang. Certificateless public key authenticated encryption with keyword search for industrial internet of things. *IEEE Transactions on Industrial Informatics*, 14(8):3618–3627, 2018.
20. A. Herzberg and H. Shulman. Oblivious and fair server-aided two-party computation. *Information Security Technical Report*, 2013.
21. Q. Huang and H. Li. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Information Sciences*, 403:1–14, 2017.
22. T. P. Jakobsen, J. B. Nielsen, and C. Orlandi. A framework for outsourcing of secure computation. In *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security*, pages 81–92. ACM, 2014.
23. A. Juels. Targeted advertising... and privacy too. In *CT-RSA*. Springer, 2001.
24. S. Kamara, P. Mohassel, and M. Raykova. Outsourcing multi-party computation. *IACR Cryptology ePrint Archive*, 2011.
25. S. Kamara, P. Mohassel, M. Raykova, and S. Sadeghian. Scaling private set intersection to billion-element sets. In *FC*. Springer, 2014.
26. S. Kamara, P. Mohassel, and B. Riva. Salus: a system for server-aided secure function evaluation. In *CCS*. ACM, 2012.
27. J. Katz and Y. Lindell. *Introduction to modern cryptography*. CRC press, 2014.
28. F. Kerschbaum. Adapting privacy-preserving computation to the service provider model. In *CSE*. IEEE, 2009.

29. F. Kerschbaum. Collusion-resistant outsourcing of private set intersection. In *Applied Computing*. ACM, 2012.
30. F. Kerschbaum. Outsourced private set intersection using homomorphic encryption. In *CCS*. ACM, 2012.
31. B. Krishnamurthy and C. E. Wills. Characterizing privacy in online social networks. In *WOSN*. ACM, 2008.
32. J. Li, X. Lin, Y. Zhang, and J. Han. Ksf-oabe: outsourced attribute-based encryption with keyword search function for cloud storage. *IEEE Transactions on Services Computing*, 10(5):715–725, 2017.
33. P. Mohassel, O. Orobets, and B. Riva. Efficient server-aided 2pc for mobile phones. *Proceedings on Privacy Enhancing Technologies*, 2016(2):82–99, 2016.
34. T. Moran and M. Naor. Split-ballot voting: everlasting privacy with distributed trust. *ACM Transactions on Information and System Security (TISSEC)*, 13(2):16, 2010.
35. A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Security and Privacy*. IEEE, 2009.
36. O. Oksuz, I. Leontiadis, S. Chen, A. Russell, Q. Tang, and B. Wang. Sevdsi: Secure, efficient and verifiable data set intersection. Technical report, Cryptology ePrint Archive, Report 2017/215.(2017). http://ia. cr/2017/215, 2017.
37. C. Patsakis, A. Zigomitros, and A. Solanas. Privacy-aware genome mining: Server-assisted protocols for private set intersection and pattern matching. In *CBMS*. IEEE, 2015.
38. K. Peng and F. Bao. A shuffling scheme with strict and strong security. In *Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference on*, pages 201–206. IEEE, 2010.
39. N. Pettersen. Applications of paillier s cryptosystem. Master's thesis, NTNU, 2016.
40. B. Pinkas, T. Schneider, and M. Zohner. Scalable private set intersection based on ot extension. *ACM Transactions on Privacy and Security (TOPS)*, 21(2):7, 2018.
41. K. Sampigethaya and R. Poovendran. A survey on mix networks and their secure applications. *Proceedings of the IEEE*, 94(12):2142–2181, 2006.
42. B. Schneier. Applied cryptography: protocols, algorithms, and source code in c. john wiley & sons, 2007.
43. B. Schoenmakers and M. Veeningen. Universally verifiable multiparty computation from threshold homomorphic cryptosystems. In *International Conference on Applied Cryptography and Network Security*, pages 3–22. Springer, 2015.
44. J. Sun, X. Zhu, and Y. Fang. A privacy-preserving scheme for online social networks with efficient revocation. In *INFOCOM*. IEEE, 2010.
45. A. Tootoonchian, S. Saroiu, Y. Ganjali, and A. Wolman. Lockr: better privacy for social networks. In *CoNEXT*. ACM, 2009.
46. V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Adnostic: Privacy preserving targeted advertising. In *NDSS*, 2010.
47. N. Volgushev, M. Schwarzkopf, B. Getchell, M. Varia, A. Lapets, and A. Bestavros. Conclave: secure multi-party computation on big data. In *European Conference on Computer Systems*, 2019.
48. Q. Zheng and S. Xu. Verifiable delegated set intersection operations on outsourced encrypted data. In *IC2E*. IEEE, 2015.
49. B. Zhu, J. Sun, J. Qin, and J. Ma. The public verifiability of public key encryption with keyword search. In *International Conference on Mobile Networks and Management*, pages 299–312. Springer, 2017.

50. X. Zou, H. Li, Y. Sui, W. Peng, and F. Li. Assurable, transparent, and mutual restraining e-voting involving multiple conflicting parties. In *INFOCOM, 2014 Proceedings IEEE*, pages 136–144. IEEE, 2014.