

Meetings through the cloud: Privacy-preserving scheduling on mobile devices

Igor Bilogrevic^{a,*}, Murtuza Jadliwala^a, Praveen Kumar^b, Sudeep Singh Walia^b, Jean-Pierre Hubaux^a, Imad Aad^c, Valteri Niemi^c

^a Laboratory for Communications and Applications 1, EPFL, CH 1015 Lausanne, Switzerland

^b Department of Computer Science and Engineering, IIT Kharagpur, India

^c Nokia Research Center, Lausanne, Switzerland

ARTICLE INFO

Article history:

Received 16 February 2011

Accepted 12 April 2011

Available online 14 May 2011

Keywords:

Activity scheduling

Mobile devices

Client-server architecture

Homomorphic encryption

ABSTRACT

Mobile devices are increasingly being used to store and manage users' personal information, as well as to access popular third-party context-based services. Very often, these applications need to determine common availabilities among a set of user schedules, in order to allow colleagues, business partners and people to meet. The privacy of the scheduling operation is paramount to the success of such applications, as often users do not want to share their personal schedule details with other users or third-parties.

In this paper, we propose practical and privacy-preserving solutions for mobile devices to the server-based scheduling problem. Our three novel algorithms take advantage of the homomorphic properties of well-known cryptosystems in order to privately and efficiently compute common user availabilities. We also formally outline the privacy requirements in such scheduling applications and we implement our solutions on real mobile devices. The experimental measurements and analytical results show that the proposed solutions not only satisfy the privacy properties but also fare better, in regard to computation and communication efficiency, compared to other well-known solutions. Finally, we assess the utility and expectations, in terms of privacy and usability, of the proposed solutions by means of a targeted survey and user-study of mobile-phone users.

© 2011 Elsevier Inc. All rights reserved.

1. Introduction

Users rely increasingly on mobile devices such as smartphones, netbooks and lightweight internet tablets to access information while on the move (Dailywireless.org, 2011), and very often they use the same equipment to store personal information about their daily schedules and activities (CHILabs PDA (Personal Digital Assistants) Use Study, 2011). Although many context and data sharing applications such as Google Maps, Facebook and Twitter are popular, activity management and synchronization applications are also gaining more and more attention (Google Smart Rescheduler, 2011). Applications such as Microsoft Outlook, Apple iCal and Nokia Ovi are available on mobile devices and they all offer time and activity management services. One desirable feature in such applications is activity *scheduling*: colleagues can schedule meetings at common available time slots, groups of friends can organize parties on weekends and people unbeknownst to each other can engage in dating based on their common free/busy hours.

One concern in such scheduling applications is that often users would prefer not to share all personal information with everyone. For example, they may only want to share common availabilities, but not details about other records. They may also have reservations about sharing personal information with third-party service providers. Therefore, privacy of personal information, *vis-à-vis* service providers and peers, is paramount for the success of such scheduling applications. For instance, a well-known service that allows users to find all common availabilities is Doodle. However, Doodle does not provide privacy: Each user and the doodle server see the free/busy state of every user, and the private information that is leaked to all users and the central server is well beyond just the common available slots. Cultural, religious and many other private information can be easily inferred from availability patterns. Even if pseudonyms are used instead of real names, the server and all peers still know what time slots are available for everyone and how many users are free or busy.

Privacy-preserving scheduling problems have been extensively studied in the past by researchers from the theoretical perspective, for instance, by modeling them as set intersection problems (Kissner and Song, 2005; De Cristofaro and Tsudik, 2010), distributed constraint satisfaction problems (Wallace and Freuder, 2005; Yokoo et al., 2005; Silaghi and Mitra, 2004; Silaghi, 2004), secure multi-party computation problems (Herlea et al., 2001; Du and Atallah, 2001) and by framing them in the e-voting con-

* Corresponding author. Tel.: +41 21 693 66 21; fax: +41 21 693 66 10.

E-mail addresses: igor.bilogrevic@epfl.ch (I. Bilogrevic),

murtuza.jadliwala@epfl.ch (M. Jadliwala), praveenkumar@cse.iitkgp.ernet.in (P. Kumar), sswalia@cse.iitkgp.ernet.in (S.S. Walia), jean-pierre.hubaux@epfl.ch (J.-P. Hubaux), imad.aad@nokia.com (I. Aad), valteri.niemi@nokia.com (V. Niemi).

text (Kellermann and Böhme, 2009). Traditionally, there are two possible approaches to the scheduling problems: distributed and centralized. Distributed solutions do not rely on a third-party provider (and thus they prevent revealing information to the provider), but have several limitations. For instance, due to the frequent and intensive message exchanges among peers, scalability and computational complexity is an issue when dealing with a large number of (resource-limited) mobile devices; moreover, the need of sequencing among peers and the unpredictability of scheduling results (if a user interrupts the protocol) are two additional drawbacks. The centralized approaches, such as cloud-based computing, are better in terms of scalability, communication cost, complexity, synchronization and resilience but usually do not provide privacy, because users are required to transmit their personal information to the provider.

Our goal is to provide simple, practical and feasible solutions to the scheduling problem which, in addition to ensuring reasonable privacy guarantees, are easily integrated with existing operational models and mobile service providers. In this paper, we follow a centralized approach for addressing the problem of efficient and privacy-preserving scheduling. In the proposed schemes, users are able to determine common time slots without revealing any other information to either the other participants or to the central scheduling server. Our specific contributions are as follows. First, by building on the work of authors in related domains, we formally define the basic privacy requirements for users in a scheduling scenario. Second, we propose three novel privacy-preserving scheduling algorithms that take advantage of the homomorphic properties of asymmetric cryptosystems. Third, we implement the proposed algorithms on a test-bed of Nokia mobile devices and perform extensive experiments in order to verify their computation and communication overheads. Moreover, we explain how the system can be further made resilient to collusion and other well-known active attacks. Finally, we present the modalities and results of a targeted user-study on mobile-phone users, focused on both privacy and usability aspects of our applications. To the best of our knowledge, we believe this is the first implementation and extensive testing of privacy-preserving scheduling schemes on commercial mobile devices.

The paper is organized as follows. We introduce the state-of-the-art in Section 2 and the system model and problem definition in Section 3. We formalize the privacy requirements for the scheduling problem in Section 4 and outline our algorithms in Section 5. We present a comparative analysis and implementation results in Section 6, and we discuss the extensions of our schemes in Section 7. We summarize the results of our user-study in Section 8, and we conclude the paper in Section 9.

2. State of the art

In the literature, the four most relevant bodies of work that address privacy in scheduling or similar scenarios are based on techniques from private set-intersection (Kissner and Song, 2005; De Cristofaro and Tsudik, 2010), distributed constraint satisfaction (Wallace and Freuder, 2005; Yokoo et al., 2005; Silaghi and Mitra, 2004; Silaghi, 2004), secure multi-party computation (Herlea et al., 2001; Du and Atallah, 2001) and e-voting (Kellermann and Böhme, 2009). Hereafter, we review the most relevant aspects of such approaches.

In the private set-intersection domain, Kissner and Song (2005) use mathematic properties of polynomials to design privacy-preserving union, intersection and element reduction operations on private multisets by leveraging on the Goldwasser–Micali homomorphic encryption scheme (Goldwasser and Micali, 1984). De Cristofaro and Tsudik (2010) provide efficient variations of

private-set intersection protocols and present a comparison in terms of computational and communication complexity, adversarial model and privacy. The authors also give informal definitions of client and server privacy. However, PSI approaches are generally distributed, and an efficient extension to an n -party protocol is challenging. In the meeting scheduling scenario, for instance, a trivial extension of the 2-party PSI to n parties (by running a 2-party protocol between each pair of users) would undermine the privacy of users' schedules as well; knowing the personal availability and the aggregate availability is sufficient to infer the other party's schedule.

Distributed constraint satisfaction approaches were investigated by Wallace and Freuder (2005): they study the tradeoff between privacy and efficiency and show that the information that entities learn during the negotiation of a common schedule has, in some cases, a tremendous impact on privacy. Details of an accept/reject response are exploited by intelligent agents in order to successfully infer the availabilities of other peers involved in the scheduling process. Similarly, Zunino and Campo (2009) design a scheduling system in which entities learn and refine their knowledge about user preferences using a Bayesian network. Yokoo et al. (2005) use secret sharing among third-party servers in order to determine a suitable agreement among entities in a collusion-resistant way.

Solutions based on secure multi-party computation were investigated in Du and Atallah (2001) and a practical scheme was proposed in Herlea et al. (2001). Herlea et al. (2001), for instance, design and evaluate a distributed secure scheduling protocol by relying on properties of the XOR operation over binary values, in which all users contribute to the secrecy of individual schedules while ensuring the correctness of the results. Although not a pure e-voting scheme, Kellermann and Böhme (2009) proposed an event scheduling protocol that inherits several security and privacy requirements from the e-voting context. However, a formal study of such properties and experimental performance results are missing in their work.

In contrast to most of the above solutions, we take a more centralized approach (with a single third-party server) for the privacy-preserving scheduling problem. Our solutions overcome communication and computational complexities intrinsic to most distributed approaches discussed above, as well as ensure that no private information (other than the resulting common availabilities) is exposed. Moreover, our protocols can easily fit into today's popular provider–consumer service architectures without incurring a huge communication cost on the service-provider.

3. System model

In this section, we outline the network and adversary model and formally define the scheduling problem.

3.1. Network model

We assume that there is a total of N users u_i , $i \in \{1 \dots N\}$, that want to schedule an activity (meeting, party) at a common available time slot. Each user has a private schedule x_i represented by a string of bits $x_i = [b_{i,1}, b_{i,2}, \dots, b_{i,m}]$, where each bit $b_{i,j} \in \{0, 1\}$ expresses the availability of user u_i in a particular time slot j ; $b_{i,j} = 1$ means that user u_i is available at time slot j , whereas $b_{i,j} = 0$ means that the user is not available.¹ We assume that the length m of x_i , i.e. the

¹ In general, however, users may assign not only a binary value (available or busy) for each time slot, but they could express preferences (Ephrati et al., 1994; Franzin et al., 2004). For example, $b_{i,j} \in 0, \dots, 10$ where $b_{i,j} = 0$ means that user u_i is busy in the time slot j , whereas its preference would increase if $b_{i,j} \geq 1$. For simplicity of

time horizon of the individual schedules, is constant for all users. The value of m can either be predecided by the participants or fixed by the application.

Moreover, we assume that each user's device is able to perform public key cryptographic operations and that there is a semi-honest (Goldreich, 2001) (as detailed in Section 3.2) third-party performing the scheduling computations. The latter must be able to communicate with the users and run public key cryptographic functions as well. For instance, a common public-key infrastructure using the RSA (Rivest et al., 1978) cryptosystem could be employed. All communications between a user and the third-party server will be encrypted with the latter's public key for the purposes of confidentiality of the schedules with respect to other users, for authentication and integrity protection. Thus, all users know the public key of the server but nobody, except the server, knows the corresponding private key. For simplicity of exposition, in our algorithms we do not explicitly show the cryptographic operations involving the server's public/private key.

We assume that the N users share a common secret, which is used to derive (i) a fresh common key pair (K_p, K_s) , where K_p is the public key and K_s is the private key, and (ii) a fresh bit permutation function $\sigma = [\sigma_1, \dots, \sigma_m]$ before initiating the scheduling operation. This could be achieved, for example, through a secure credential establishment protocol (Cachin and Strobl, 2004; Chen et al., 2008; Lin et al., 2009). Thus, these keys and permutations are derived and known to each member of the group but not to the server. We refer to the encryption of a message M with the group public key as $E_{K_p, r}(M) = C$, where r is a random integer that is eventually needed, and to the decryption of the encrypted message C as $D_{K_s}(C) = M$. The permutation σ , although not strictly required, is used in order to randomize the order of bits sent to the server. This prevents the server from gaining any knowledge about which time slot is being evaluated in each computation.

3.2. Adversarial model

3.2.1. Server

The third-party server is assumed to execute the scheduling protocols correctly, but it tries to learn any information it can from the input it gets by the users and the computations it performs. The server can accumulate the knowledge about users in each computation it performs. We refer to this adversarial behavior as *semi-honest*. In most practical settings, where service providers have a commercial interest in providing a faithful service to their customers, the assumption of a semi-honest server is generally sufficient. More details about the semi-honest model can be found in Goldreich (2001).

3.2.2. Users

Users also want to learn private information about other users' schedules and, in addition to the passive eavesdropping attacks, users could act maliciously by generating fake users, manipulating their own schedules or by colluding with other users or the scheduling server. Initially, we assume that users are honest but curious (or semi-honest), and afterwards we present more active (or malicious) types of user adversaries in Section 7.2.

Although, as mentioned, the semi-honest adversarial model is sufficient in most practical settings, considering the commercial interest of service providers and the mutual trust among participants, it does not include possible malicious behavior by the server or users. For instance, the server could collude with the participants or generate fake participants in order to obtain private information

of the participants. Similarly, users might collude with other users or try to maliciously modify their schedules in order to disrupt the execution of the protocol or to gain information about other users' schedules. We address such active attacks by both users and server in Section 7.2, and we describe how such attacks can be thwarted using existing cryptographic mechanisms.

3.3. Centralized scheduling algorithm

Given a group of N users $u_i, i \in \{1 \dots N\}$, each with private schedules $x_i = [b_{i,1}, \dots, b_{i,m}]$, the scheduling problem is to find time slots j such that $\forall i = 1 \dots N, b_{i,j} = 1$, i.e. all users are available in the same time slot j . We refer to an algorithm that solves the scheduling problem as a *scheduling algorithm*. Fig. 1 shows a functional diagram of a generic privacy-preserving scheduling protocol, where the scheduling algorithm A is executed by a server. Formally, a scheduling algorithm A accepts the following inputs and produces the respective outputs:

- Input: a transformation of individual schedules

$$f(b_{i,1}, \dots, b_{i,m}), \quad \forall i = 1 \dots N.$$

where f is a one-way public transformation function (based on secret key) such that it is hard (success with only a negligible probability) to determine the input of the function without knowing the secret key, just by observing the output.

- Output: a function $f(Y), Y = y^1, \dots, y^j, \dots, y^m$ where:

$$y^j = \begin{cases} YES & \text{if } b_{i,j} = 1, \quad \forall i = 1 \dots N \\ NO & \text{otherwise} \end{cases}$$

such that each user is able to compute $Y = f^{-1}(f(Y))$ using its local data. As we will see later on, we use the well-known cryptosystems ElGamal (1985), Paillier (1999) and Goldwasser–Micali (Goldwasser and Micali, 1984) as our transformation and output functions f .

A centralized scheduling process works as follows. Each user $u_i, i \in \{1 \dots N\}$ computes $f_i = f(b_{i,1}, \dots, b_{i,m})$ and sends it to the third-party server, which then executes the scheduling algorithm A on the received inputs $f_i, \forall i$, and produces $f(Y) = A(f_1, \dots, f_N)$. Finally, the server sends $f(Y)$ to each user who then obtains $Y = f^{-1}(f(Y))$. Fig. 2 shows one execution of such a generic centralized scheduling process.

4. Privacy definitions

As mentioned earlier, in this paper we follow a centralized approach to solve the privacy-preserving scheduling problem. In other words, we assume that a third-party, given users' individual private schedules, computes their common availabilities (time slots). The privacy provided by a centralized scheduling algorithm can be defined in terms of the following two components: a) User-privacy and b) Server-privacy. Hereafter, we formally define each of these components. The symbols used throughout the paper are summarized in Table 1.

4.1. User-privacy

The *user-privacy* of any centralized scheduling algorithm A measures the probabilistic advantage that any user $u_i, i \in \{1 \dots N\}$ gains towards learning the private schedules of at least one other user $u_j, j \neq i$, except their common availabilities, after all users have participated in the execution of the algorithm A . In order to accurately

exposition, we assume a binary value here. We later discuss a more general case with non-binary costs in Section 7.

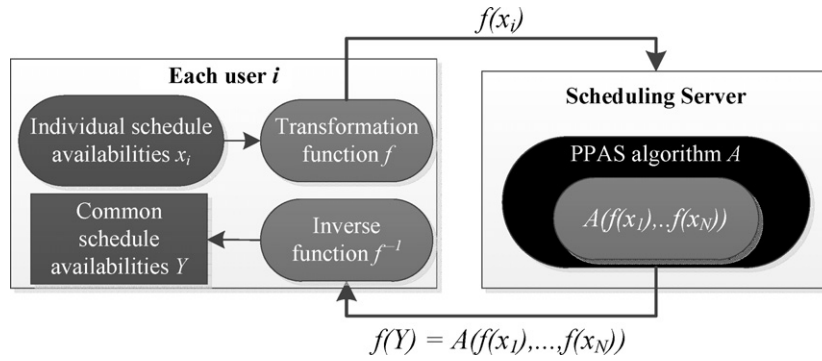


Fig. 1. Functional diagram of the privacy-preserving activity scheduling protocol, where each user sends his own transformed schedule availabilities to the scheduling server and obtains the aggregate availabilities. The scheduling obviously performs the aggregated availabilities, without knowing the individual user schedules.

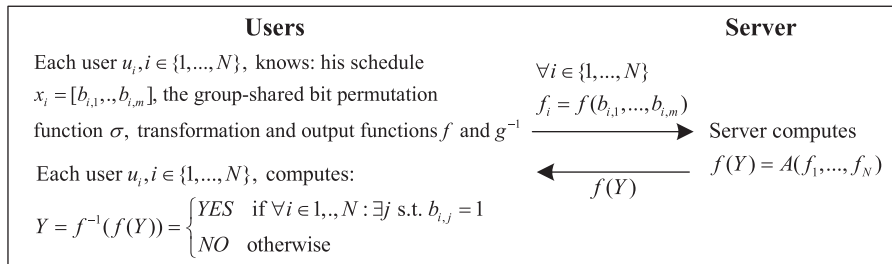


Fig. 2. A generic scheduling protocol. Users first send their transformed schedules f_i to the server, which then performs the scheduling algorithm A on the received data and sends the encrypted output $f(Y)$ back to each user.

measure users' privacy, we need to compute the following two advantages. First, we measure the *Identifiability Advantage*, which is the probabilistic advantage of an adversary in correctly guessing a schedule bit (which is not a common availability) of any other user. We denote it as $Adv_{u_i}^{IDT}(A)$. Second, we measure the *Linkability Advantage*, which is the probabilistic advantage of an adversary in correctly guessing that any two or more other users have exactly the same corresponding schedule bit (not a common availability bit) without necessarily knowing the values of those bits. We denote this advantage as $Adv_{u_i}^{LNK}(A)$. We make the following straightforward observation.

Observation 1. If an adversary has identifiability advantage over two corresponding schedule bits of two different users, this implies that it has linkability advantage over those two bits as well. However, the inverse is not necessarily true.

We semantically define the identifiability and linkability advantages using a challenge-response methodology. Challenge-response games have been widely used in cryptography to prove the security of cryptographic protocols. We now describe

such a challenge-response game for the identifiability advantage $Adv_{u_i}^{IDT}(A)$ of any user u_i participating in the algorithm A as follows.

1. Initialization: Challenger privately collects $x_i = [b_{i,1}, \dots, b_{i,m}]$ and $f_i = f(b_{i,1}, \dots, b_{i,m})$ from all users $u_i, i \in \{1 \dots N\}$.
2. Scheduling: Challenger computes $f(Y) = A(f_1, f_2, \dots, f_N)$ with the users and sends $f(Y)$ to all users u_1, u_2, \dots, u_N .
3. Challenger randomly picks a user $u_i, i \in \{1 \dots N\}$, as the adversary.
4. u_i picks $j \in \{1 \dots N\}$, s.t. $j \neq i$ and sends it to the challenger.
5. Challenge: the challenger picks a random time slot $p \in \{1 \dots m\}$, s.t., $\exists b_{k,p} = 0$ for at least one $k \in 1, \dots, N$. Challenger then sends (j, p) to the user u_i . This is the challenge.
6. Guess: User u_i sends $b'_{j,p} \in \{0, 1\}$ to the challenger as a response to his challenge. If $b'_{j,p} = b_{j,p}$, the user u_i (adversary) wins; otherwise, he loses.

The identifiability advantage $Adv_{u_i}^{IDT}(A)$ can be defined as

$$Adv_{u_i}^{IDT}(A) = \left| Pr_{u_i}[b'_{j,p} = b_{j,p}] - \frac{1}{2} \right| \tag{1}$$

where $Pr_{u_i}[b'_{j,p} = b_{j,p}]$ is the probability of user u_i winning the game (correctly answering the challenge in the challenge-response game), computed over the coin flips of the challenger, $b'_{j,p}$ is u_i 's guess about the schedule of user u_j in the time slot p and $b_{j,p}$ is u_j 's true availability. An external attacker, having no access to the output of the algorithm, has obviously no advantage at all. Thus, we focus on the non-trivial case with participating users only.

Similarly, we describe the challenge-response game for the linkability advantage $Adv_{u_i}^{LNK}(A)$ of any user u_i as follows.

1. Initialization: Challenger privately collects $x_i = [b_{i,1}, \dots, b_{i,m}]$ and $f_i = f(b_{i,1}, \dots, b_{i,m})$ from all users $u_i, i \in \{1 \dots N\}$.
2. Scheduling: Challenger computes $f(Y) = A(f_1, f_2, \dots, f_N)$ with the users and sends $f(Y)$ to all users u_1, u_2, \dots, u_N .

Table 1
Table of symbols.

Symbol	Definition
$Adv^{LNK}(A)$	Linkability advantage
$Adv^{IDT}(A)$	Identifiability advantage
$D(C)$	Decryption of a ciphertext C
$E_{K,r}(m)$	Encryption of a message m using the key K and a random number r
K_P	Shared public key of the N users
K_S	Shared private key of the N users
m	Number of slots of each individual schedule
N	Number of users
$x_i = [b_i, 1, \dots, b_{i,m}]$	Schedule of user u_i , where $b_{i,j}$ is the availability at time slot j
$\sigma = \sigma_1, \dots, \sigma_m$	Schedule permutation function

3. Challenger randomly picks a user u_i , $i \in \{1 \dots N\}$, as the adversary.
4. u_i picks $h, j \in \{1 \dots N\}$, s.t. $j \neq h$, $j \neq i$, $h \neq i$ and sends (h, j) to the challenger.
5. Challenge: Challenger randomly picks a time slot $p \in \{1 \dots m\}$, s.t., $\exists b_{k,p} = 0$ for at least one $k \in 1, \dots, N$. Challenger then sends (j, p) and (h, p) to the user u_i . This is the challenge.
6. Guess: User u_i decides if $b_{j,p} = b_{h,p}$ or not. User u_i sets $b' = 1$ if he decides $b_{j,p} = b_{h,p}$ and $b' = 0$ if he decides $b_{j,p} \neq b_{h,p}$. User u_i sends b' to the challenger as a response to his challenge. If $b_{j,p} = b_{h,p}$ and $b' = 1$ or if $b_{j,p} \neq b_{h,p}$ and $b' = 0$, the user u_i (adversary) wins; otherwise, he loses.

The linkability advantage $Adv_{u_i}^{LNK}(A)$ can be defined as

$$Adv_{u_i}^{LNK}(A) = \left| Pr_{u_i}[(b_{j,p} = b_{h,p}) \wedge b' = 1] \vee ((b_{j,p} \neq b_{h,p}) \wedge b' = 0) - \frac{1}{2} \right|$$

where $Pr_{u_i}[\cdot]$ is the probability of user u_i winning the game, computed over the coin flips of the challenger. As for the identifiability advantage, an external attacker has no linkability advantage at all.

We now define the user-privacy of the scheduling algorithm A on a per-execution basis as follows:

Definition 1. An execution of the centralized scheduling algorithm A is *user-private* if both the identifiability advantage $Adv_{u_i}^{IDT}(A)$ and the linkability advantage $Adv_{u_i}^{LNK}(A)$ of each participating user u_i , $i \in \{1, \dots, N\}$ is negligible.

A function $f(x)$ is called *negligible* if, for any positive polynomial $p(x)$, there is an integer B such that for any integer $x > B$, $f(x) < 1/p(x)$ (Goldreich, 2001).

Definition 1 says that a particular execution of the scheduling algorithm is user-private if and only if users do not gain any (actually, negligible) additional knowledge about the schedule bits of any other user, except the schedule bits that have a value 1 for all users (common availabilities).

4.1.1. Server-privacy

The *server-privacy* of any (centralized) scheduling algorithm A measures the probabilistic advantage that the server (which executes the scheduling algorithm A and observes the inputs from the users) gains towards learning the private schedules of at least one user u_i , $i \in \{1 \dots N\}$. As in the case of user-privacy, we need to compute the following two advantages. First, the advantage of the server in guessing correctly any schedule bit of any user participating in the scheduling algorithm, called as *Identifiability Advantage* and denoted as $Adv_S^{IDT}(A)$. Second, the advantage of the server in guessing correctly that any two (or more) participating users have exactly the same corresponding schedule bits without necessarily knowing the values of those bits, called the *Linkability Advantage* and denoted as $Adv_S^{LNK}(A)$.

The server identifiability and linkability advantages are defined in a similar fashion as the user advantages. The challenge-response game for the server identifiability advantage $Adv_S^{IDT}(A)$ is defined as follows.

1. Initialization: Challenger privately collects $x_i = [b_{i,1}, \dots, b_{i,m}]$ and the server privately collects $f_i = f(b_{i,1}, \dots, b_{i,m})$ from all users u_i , $i \in \{1 \dots N\}$.
2. Scheduling: Server computes $f(Y) = A(f_1, f_2, \dots, f_N)$ with the users and sends $f(Y)$ to all users u_1, u_2, \dots, u_N .
3. Server picks $i \in \{1 \dots N\}$ and sends it to the challenger.
4. Challenge: Challenger randomly picks a time slot $p \in \{1 \dots m\}$. Challenger then sends (i, p) to the server. This is the challenge.

5. Guess: server sends $b'_{i,p} \in \{0, 1\}$ to the challenger as a response to his challenge. If $b'_{i,p} = b_{i,p}$, the server (adversary) wins; otherwise, he loses.

The identifiability advantage $Adv_S^{IDT}(A)$ is defined as

$$Adv_S^{IDT}(A) = \left| Pr_S[b'_{i,p} = b_{i,p}] - \frac{1}{2} \right| \quad (2)$$

where $Pr_S[b'_{i,p} = b_{i,p}]$ is the probability of the server winning the game, computed over the coin flips of the challenger.

The challenge-response game for the server linkability advantage $Adv_S^{LNK}(A)$ is defined as follows.

1. Initialization: Challenger privately collects $x_i = [b_{i,1}, \dots, b_{i,m}]$ and the server privately collects $f_i = f(b_{i,1}, \dots, b_{i,m})$ from all users u_i , $i \in \{1 \dots N\}$.
2. Scheduling: Server computes $f(Y) = A(f_1, f_2, \dots, f_N)$ with the users and sends $f(Y)$ to all users u_1, u_2, \dots, u_N .
3. Server picks $h, j \in \{1 \dots N\}$, s.t. $j \neq h$ and sends (h, j) to the challenger.
4. Challenge: Challenger randomly picks $p \in \{1 \dots m\}$ and then sends (j, p) and (h, p) to the server. This is the challenge.
5. Guess: Server decides if $b_{j,p} = b_{h,p}$ or not. Server sets $b' = 1$ if he decides $b_{j,p} = b_{h,p}$ and $b' = 0$ if he decides $b_{j,p} \neq b_{h,p}$. Server sends b' to the challenger as a response to his challenge. If $b_{j,p} = b_{h,p}$ and $b' = 1$ or if $b_{j,p} \neq b_{h,p}$ and $b' = 0$, the server (adversary) wins; otherwise, he loses.

The linkability advantage $Adv_S^{LNK}(A)$ is defined as

$$Adv_S^{LNK}(A) = \left| Pr_S[(b_{j,p} = b_{h,p}) \wedge b' = 1] \vee (b_{j,p} \neq b_{h,p}) \wedge b' = 0 - \frac{1}{2} \right| \quad (3)$$

where $Pr_S[\cdot]$ is the probability of the server winning the game, computed over the coin flips of the challenger.

The server-privacy of the scheduling algorithm A on a per-execution basis can then be defined as follows:

Definition 2. An execution of the centralized scheduling algorithm A is *server-private* if both the identifiability advantage $Adv_S^{IDT}(A)$ and the linkability advantage $Adv_S^{LNK}(A)$ of the server is negligible.

Now, it is reasonable to assume that in practice users will be able to perform multiple executions of the scheduling algorithm with possibly different participating sets of users. This is especially true if such an algorithm is offered, for example, as a service by mobile service providers to their subscribers. Thus, privacy of the scheduling algorithm should be defined over multiple executions. First, we define a *private execution* as follows:

Definition 3. A *private execution* is an execution which does not reveal more information than what can be derived from its result and the prior knowledge.

Based on how memory is retained over sequential executions, we define two types of algorithm executions, namely, independent and dependent:

Definition 4. An *independent* (respectively, *dependent*) execution is a single private execution of the scheduling algorithm defined in Section 3.3 in which *no* (respectively, *some*) information of an earlier and current execution is retained and passed to a future execution.

The information retained can include past inputs to the algorithm, intermediate results (on the server) and the outputs of the algorithm. Based on the type of executions, we define a privacy-preserving scheduling algorithm as follows:

Definition 5. A scheduling algorithm A is *execution* (respectively *fully*) *privacy-preserving* if and only if for every *independent* (respectively *all*) execution(s):

1. A is correct; All users are correctly able to compute $y^j = 1, \forall j = 1 \dots m$ if and only if $b_{i,j} = 1, \forall i = 1 \dots N$.
2. A is user-private in every execution.
3. A is server-private in every execution.

A fully privacy-preserving algorithm is a much stronger (and difficult to achieve) privacy requirement. In this work, similar to earlier efforts, we focus on achieving execution privacy. The following observation gives the relationship between fully privacy-preserving and execution privacy-preserving scheduling algorithms.

Observation 2. Any scheduling algorithm A , as defined in Section 3.3, is execution privacy-preserving if it is fully privacy-preserving. However, the inverse is not true.

Next, we outline our centralized scheduling algorithms.

5. Privacy-preserving scheduling algorithms

In this section, we present our three privacy-preserving scheduling algorithms. For each algorithm, we first outline the basic cryptographic properties that are used, and then we describe and show their operational mechanisms in detail. We finally state the privacy guarantees provided by each of the algorithms.

5.1. SchedELG

Our first privacy-preserving centralized scheduling scheme is based on the ElGamal (1985) cryptosystem. The security of the ElGamal encryption relies on the intractability of the discrete logarithm problem (DLP), which assumes that it is computationally infeasible to obtain the private key K_s given the public key (g, h) , where g is a generator of a multiplicative cyclic group G of prime order q and $h = g^{K_s} \bmod q$.

Our protocol *SchedELG* uses the *homomorphic* property of the ElGamal cryptosystem in order to allow the scheduling server to compute the aggregated availabilities by working only on the encrypted individual schedules. For instance, it can be verified that the ElGamal scheme satisfies:

$$\begin{aligned} D(E_{K_p, r_1}(m_1) \cdot E_{K_p, r_2}(m_2)) &= D((g^{r_1}, m_1 h^{r_1}) \cdot (g^{r_2}, m_2 h^{r_2})) \\ &= D(g^r, (m_1 \cdot m_2) h^r) = m_1 \cdot m_2 \end{aligned}$$

where $r = r_1 + r_2 \in \mathbb{Z}_q$ is a random integer. Moreover, being a probabilistic encryption scheme, it follows that if $r_1 \neq r_2, E_{K_p, r_1}(m) \neq E_{K_p, r_2}(m)$.

For the *SchedELG* algorithm, we assume that the meeting participants represent their availabilities in the following way: $b_{i,j}^* = 1$ if $b_{i,j} = 1$, but $b_{i,j}^* = R$ (where $R \in \mathbb{Z}_q, R > 1$ is a random integer) if $b_{i,j} = 0$.

5.1.1. Scheme

The privacy-preserving scheduling protocol *SchedELG* is shown in Fig. 3. All users first select the sequence of time slots according to the permutation σ , i.e., $\sigma_j, \forall j = 1 \dots m$, and then encrypt individually the corresponding schedule availabilities, i.e., $E_i = [E_{i, \sigma_1}, \dots, E_{i, \sigma_m}]$ where $E_{i, \sigma_j} = E_{K_p, r_{i,j}}(b_{i, \sigma_j}^*)$. Then, each user sends its E_i privately to the scheduling server that performs the multiplication $\prod_{i=1}^N E_{i, \sigma_j}$ of all users' encrypted schedules E_{i, σ_j} , for $j = 1, \dots, m$. The results of such operation are the (encrypted) aggregated availabilities of all

users for each time slot j . Next, the server replies with the aggregated encrypted result E_{sched} back to each user. Each slot in E_{sched} contains a product of the individual time-slot bits encrypted with the users' common session key. Finally, each user decrypts the result and obtains the aggregated availabilities $[y^1 = B_{\sigma_1}^*, \dots, y^m = B_{\sigma_m}^*]$ of all users u_i for each time slot σ_j . If $B_{\sigma_j}^* = 1$, it means that all users are available at time slot σ_j ; if $B_{\sigma_j}^* > 1$, then at least one user is not available and therefore σ_j is not a suitable time slot. The following result shows the correctness and privacy properties of *SchedELG*.

Lemma 1. *The protocol SchedELG is correct and execution privacy-preserving.*

Proof. Correctness: From Section 3.3, we know that any scheduling algorithm should output $f(Y)$, on inputs f_1, f_2, \dots, f_N , where $f_i = f(b_{1,1}, \dots, b_{i,m})$, such that each user is able to privately compute $Y = f^{-1}(f(Y))$, where $Y = y^1, \dots, y^j, \dots, y^m$. The output bit $y^j, \forall j$ should be such that it should take some value v if and only if all users are available. Otherwise, the output bit y^j never takes value v and should take some other value, indicating that at least one user is not available. From Fig. 3, we can see that, provided the homomorphic properties of the ElGamal cryptosystem are correct, we have that (with overwhelming probability) $y^j = 1$ if and only if $b_{i,j} = 1, \forall i$, i.e., all users are available. Otherwise we have $y^j = R$, where $R > 1$ is some random number. Thus, *SchedELG* is correct.

Privacy: In order to be *user-private*, the identifiability and linkability advantages defined in Section 3 must be a negligible function. Formally, we need that

$$\begin{aligned} Adv_{u_i}^{IDT}(SchedELG) &= \left| Pr_{u_i}[b'_{j,p} = b_{j,p}] - \frac{1}{2} \right| < \frac{1}{p(N)} \\ Adv_{u_i}^{LNK}(SchedELG) &= \left| Pr_{u_i}[(b_{j,p} = b_{h,p}) \wedge b' = 1] \vee (b_{j,p} \neq b_{h,p}) \wedge b' = 0 \right| - \frac{1}{2} < \frac{1}{p(N)} \end{aligned}$$

where $Pr_{u_i}[b'_{j,p} = b_{j,p}]$ and $Pr_{u_i}[(b_{j,p} = b_{h,p}) \wedge b' = 1] \vee (b_{j,p} \neq b_{h,p}) \wedge b' = 0$ are the probabilities of a user u_i winning the challenge-response games, and $p(N)$ is any positive polynomial function of N . Without loss of generality, we assume that the Challenger chooses user u_1 as the Adversary. Moreover, as the computation of the availabilities for all time slots are identical, we provide the proof for one time slot p only.

Hereafter we provide the privacy proofs for both client- and server-privacy, by computing the respective identifiability and linkability advantages.

• User identifiability advantage

After Step 4 of the challenger-response game, u_1 knows (i) its own schedule bit $b_{1,p}$ and (ii) the non-trivial result of the algorithm $B_p^* = b_{1,p}^* \cdot \dots \cdot b_{N,p}^* > 1$, i.e. there is at least one user that is not available in the time slot p . Therefore, the identifiability advantage becomes

$$Adv_{u_i}^{IDT}(SchedELG) = \left| Pr_{u_i}[b'_{j,p} = b_{j,p} | B_p^* > 1, b_{1,p}] - \frac{1}{2} \right|$$

where

$$\begin{aligned} &Pr_{u_i}[b'_{j,p} = b_{j,p} | B_p^* > 1, b_{1,p}] \\ &= \sum_{k=0}^1 Pr(b'_{j,p} = b_{j,p} | B_p^* > 1, b_{1,p} = k) \cdot Pr(b_{1,p} = k | B_p^* > 1) \\ &= \sum_{k=0}^1 \sum_{z=0}^1 Pr(b'_{j,p} = z \wedge b_{j,p} = z | B_p^* > 1, b_{1,p} = k) \cdot Pr(b_{1,p} = k | B_p^* > 1) \\ &= \sum_{k=0}^1 \sum_{z=0}^1 Pr(b'_{j,p} = z | B_p^* > 1, b_{1,p} = k) \cdot Pr(b_{j,p} = z | B_p^* > 1, b_{1,p} = k) \\ &\quad \cdot Pr(b_{1,p} = k | B_p^* > 1) \end{aligned}$$

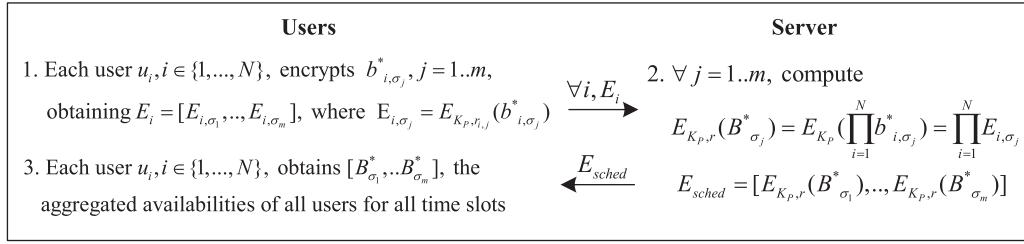


Fig. 3. SchedElg protocol.

Given that the Challenger chooses a time slot p where $\exists b_{q,p} = 0$, $q \in \{1, \dots, N\}$, we have

$$\begin{aligned} \Pr(b'_{j,p} = 0 | B_p^* > 1, b_{1,p} = 0) &= \Pr(b'_{j,p} = 1 | B_p^* > 1, b_{1,p} = 0) = 1/2 \\ \Pr(b_{j,p} = 0 | B_p^* > 1, b_{1,p} = 0) &= \Pr(b_{j,p} = 1 | B_p^* > 1, b_{1,p} = 0) = 1/2 \\ \Pr(b'_{j,p} = 0 | B_p^* > 1, b_{1,p} = 1) &= \Pr(b_{j,p} = 0 | B_p^* > 1, b_{1,p} = 1) \\ &= \frac{\sum_{m=1}^{N-1} C_m^{N-1} \cdot m}{2^{N-1} \cdot (N-1)} = \frac{2^{N-2}}{2^{N-1} - 1} \end{aligned}$$

$$\begin{aligned} \Pr(b'_{j,p} = 1 | B_p^* > 1, b_{1,p} = 1) &= \Pr(b_{j,p} = 1 | B_p^* > 1, b_{1,p} = 1) \\ &= \frac{\sum_{m=0}^{N-2} C_m^{N-1} \cdot m}{2^{N-1} \cdot (N-1)} = \frac{2^{N-2} - 1}{2^{N-1} - 1} \end{aligned}$$

which implies

$$\Pr_{u_i}[b'_{j,p} = b_{j,p} | B_p^* > 1, b_{1,p}] = \frac{a}{2} + (1-a) \cdot \frac{2^{2(N-2)} + (2^{N-2} - 1)^2}{(2^{N-1} - 1)^2}$$

where $a = \Pr(b_{1,p} = 0 | B_p^* > 1)$. By including this result, we have that

$$\text{Adv}_{u_i}^{\text{DT}}(\text{SchedElg}, N) = \left| \frac{a}{2} + (1-a) \cdot \underbrace{\frac{2^{2(N-2)} + (2^{N-2} - 1)^2}{(2^{N-1} - 1)^2}}_{\gamma} - \frac{1}{2} \right|$$

where

$$\begin{aligned} \gamma &= \frac{2^{2N-4} + 2^{2N-4} - 2 \cdot 2^{N-2} + 1}{2^{N-2} - 2 \cdot 2^{N-1} + 1} = \frac{2^{2N-3} - 2^{N-1} + 1}{2^{2N-2} - 2^N + 1} \\ &= \frac{(2^{2N-2} - 2^N + 1) + 1}{2 \cdot (2^{2N-2} - 2^N + 1)} = \frac{1}{2} + \frac{1}{2(2^{2N-2} - 2^N + 1)} \end{aligned}$$

By combining the previous expressions, we obtain

$$\begin{aligned} \text{Adv}_{u_i}^{\text{DT}}(\text{SchedElg}, N) &= \left| \frac{a}{2} + (1-a) \cdot \gamma - \frac{1}{2} \right| = \left| \frac{1-a}{2^{2N-1} - 2^{N+1} + 2} \right| \\ &= \left| \frac{1-a}{2^{N+1}(2^{N-2} - 1) + 2} \right| \stackrel{N > 2}{<} \frac{1}{2^N} \end{aligned}$$

which holds $\forall 0 \leq a \leq 1$. Therefore $\text{Adv}_{u_i}^{\text{DT}}(\text{SchedElg}, N)$ is a negligible function of the number of participants N , as it approaches zero faster than the reciprocal of any polynomial, for large enough N (Bellare, 2008).

- User linkability advantage
By definition we have

$$\text{Adv}_{u_i}^{\text{LNK}}(\text{SchedElg})$$

$$= \left| \Pr_{u_i}[(b_{j,p} = b_{n,p}) \wedge b' = 1] \vee (b_{j,p} \neq b_{n,p}) \wedge b' = 0 \mid B_p^* > 1, b_{1,p}] - \frac{1}{2} \right|$$

From the above, we obtain

$$\begin{aligned} \Pr_{u_i}[(b_{j,p} = b_{n,p}) \wedge b' = 1] \vee (b_{j,p} \neq b_{n,p}) \wedge b' = 0 \mid B_p^* > 1, b_{1,p}] \\ = \sum_{k=0}^1 \Pr[(b_{j,p} = b_{n,p}) \wedge b' = 1 \mid B_p^* > 1, b_{1,p} = k] \cdot \Pr(b_{1,p} = k \mid B_p^* > 1) \\ + \sum_{k=0}^1 \Pr[(b_{j,p} \neq b_{n,p}) \wedge b' = 0 \mid B_p^* > 1, b_{1,p} = k] \cdot \Pr(b_{1,p} = k \mid B_p^* > 1) \end{aligned}$$

which implies

$$\begin{aligned} \Pr_{u_i}[(b_{j,p} = b_{n,p}) \wedge b' = 1] \vee (b_{j,p} \neq b_{n,p}) \wedge b' = 0 \mid B_p^* > 1, b_{1,p}] &= \frac{a}{2} + (1-a) \\ \cdot \left\{ \left[\left(\frac{2^{N-2}}{2^{N-1} - 1} \right)^2 + \frac{2^{N-3} - 1}{2^{N-2}} \cdot \frac{2^{N-2} - 1}{2^{N-1} - 1} \right]^2 + \left[\frac{1}{4} + \frac{1}{2} \frac{2^{N-2} - 1}{2^{N-1} - 1} \right]^2 \right\} \end{aligned}$$

where $a = \Pr(b_{1,p} = 0 | B_p^* > 1)$. Similarly to the identifiability advantage, it can be shown that $\text{Adv}_{u_i}^{\text{LNK}}(\text{SchedElg}, N)$ is a negligible function of the number of participants N . As both identifiability and linkability advantages are negligible functions (in the number of participants N), SchedElg is user-private.

- Server advantages

The server that is performing the computations on the encrypted schedules does not know any user's schedule bit, as all schedules have been encrypted by the users prior to being sent to the server with the users' shared public key, and only they know the corresponding private key. Therefore, $\text{Adv}_S^{\text{DT}}(\text{SchedElg}) = \text{Adv}_S^{\text{LNK}}(\text{SchedElg}) = 0$, i.e. SchedElg is server-private.

□

For illustration purposes, in Fig. 4 we plotted the identifiability and linkability advantages of an adversary for SchedElg, compared

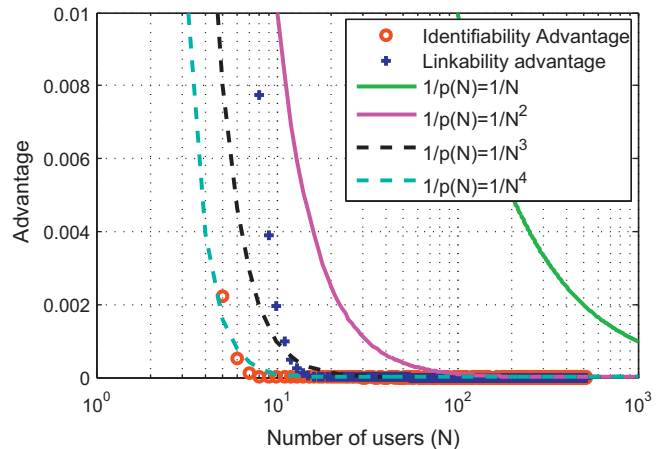


Fig. 4. Identifiability and linkability advantages of an adversary.

with polynomially (in terms of the number of participants N) decreasing functions $1/p(N)$. As confirmed by our analysis, the plot shows that both identifiability and linkability advantages are lower than the considered polynomials $1/p(N)$, for the given values of N .

5.2. SchedPa algorithm

In this section, we define our second privacy-preserving scheduling scheme, which is based on the Paillier cryptosystem (Paillier, 1999). The security of the Paillier encryption scheme is based on the intractability of determining whether an integer r is an n -residue mod n^2 , where n is a composite number. In our protocol, we use the homomorphic properties of the Paillier cryptosystem to compute in a privacy-preserving fashion the availability of all users involved in the scheduling process. In particular, one can verify that the Paillier scheme satisfies the following:

$$\begin{aligned} D[E_{K_p, r_1}(m_1) \cdot E_{K_p, r_1}(m_2) \bmod n^2] &= m_1 + m_2 \bmod n \\ D[E_{K_p, r}(m_1)^{m_2} \bmod n^2] &= m_1 \cdot m_2 \bmod n \end{aligned}$$

where $r_i, r \in \mathbb{Z}_n^*$ are random numbers chosen by the encrypters, $m \in \mathbb{Z}_n$ is the message to encrypt and $n=pq$ where p, q are two large primes. The randomness in the encryption ensures that if $r_1 \neq r_2, E_{K_p, r_1}(m) \neq E_{K_p, r_2}(m)$.

To adapt our scheme to the addition property of Paillier's homomorphism, we take the bit value $\bar{b}_{i,j}$ in the computation instead of the original bit value $b_{i,j}$ as follows: $\bar{b}_{i,j} = 0$ if $b_{i,j} = 1$, and $\bar{b}_{i,j} = r$ (where $r \in \mathbb{Z}_n^*, r > 1$ is a random integer) if $b_{i,j} = 0$.

5.2.1. Scheme

The corresponding privacy-preserving scheduling protocol is shown in Fig. 5. First, all users select the sequence of time slots according to the permutation σ , i.e., $\sigma_j, \forall j=1, \dots, m$, and then encrypt individually the corresponding availabilities, i.e. $E_i = [E_{i, \sigma_1}, \dots, E_{i, \sigma_m}]$ where $E_{i, \sigma_j} = E_{K_p, r_{i,j}}(\bar{b}_{i, \sigma_j})$. Then, each user sends its E_i privately to the scheduling server that performs the multiplication and exponentiation ($\prod_{i=1}^N E_{i, \sigma_j}$)^R of all users' encrypted schedules E_{i, σ_j} , for $j=1, \dots, m$, in order to obtain the encryption of the value V_{σ_j} that is needed by the users. Afterwards, the server sends the aggregated encrypted result E_{sched} back to each user. Each slot in E_{sched} contains a randomly scaled sum of the individual time-slot bits \bar{b}_{i, σ_j} encrypted with the users' common session key. Finally, each user decrypts the result and knows that if $V_{\sigma_j} = 0$, the time slot σ_j is available for everybody. If $V_{\sigma_j} > 1$, then at least one user is not available. Note that even if the server chooses $R=1$, the privacy of the users is preserved with $\bar{b}_{i,j}$. The following result shows the correctness and privacy properties of SchedPa.

Lemma 2. *The protocol SchedPa is correct and execution privacy-preserving.*

Proof. Correctness: From Section 3.3, we know that any scheduling algorithm should output $f(Y)$, on inputs f_1, f_2, \dots, f_N , where $f_i = f(b_{i,1}, \dots, b_{i,m})$, such that each user is able to privately compute $Y = f^{-1}(f(Y))$, where $Y = y^1, \dots, y^j, \dots, y^m$. The output bit $y^j, \forall j$ should be such that it should take some value v if and only if all users are available. Otherwise, the output bit y^j never takes value v and should take some other value, indicating that at least one user is not available. From Fig. 3, we can see that, provided the homomorphic properties of the Paillier cryptosystem are correct, we have that (with overwhelming probability) $y^j = 0$ if and only if $b_{i,j} = 1, \forall i$, i.e., all users are available. The value of $y^j = R$, where $R > 1$ is some random number, otherwise. Thus, SchedPa is correct.

Privacy: Hereafter we present the privacy proofs, both for user- and server-privacy.

- User advantages

The knowledge that any user u_i has in the SchedPa game is the same as in SchedElG. In particular, u_i knows that $V_p = R \cdot \sum_{k=1}^N b_{k,p} > 0$ and therefore it knows that there is at least one user $u_k, k \in \{1, \dots, N\}$ that is not available in the time slot p . Moreover, each user u_i knows its own schedule $b_{i,p}$. As a consequence, $Adv_{u_i}^{DT}(\text{SchedPa}) = Adv_{u_i}^{DT}(\text{SchedElG})$ and $Adv_{u_i}^{LNK}(\text{SchedPa}) = Adv_{u_i}^{LNK}(\text{SchedElG})$ and therefore SchedPa is user-private.

- Server advantages

As in the SchedElG algorithm, the server performing the SchedPa algorithm does not have access to any schedule bit and therefore SchedPa is server-private.

□

5.3. SchedGM algorithm

In this section, we present our third privacy-preserving scheduling algorithm, which is based on the Goldwasser–Micali (GM) cryptographic scheme (Goldwasser and Micali, 1984). The security of the GM encryption relies on the intractability of the quadratic residuosity problem, i.e. on the infeasibility of determining whether or not an integer r is a quadratic residue mod n when the Jacobi symbol for r is 1, given $n=pq$ where p, q are large primes. SchedGM makes use of the following homomorphic property of the GM cryptosystem:

$$D[E_{K_p, r_1}(m_1) \cdot E_{K_p, r_2}(m_2)] = m_1 \oplus m_2$$

The intuition behind the protocol is based on the work by Herlea et al. (2001), in which users privately establish a global bit mask (unknown to any user) and then compare all the masked availabilities without knowing the true bit value b_{i, σ_j} of the other users. If all users have the same masked bit value for a given time slot σ_j , then each user knows that everybody else has the same availability, which can be inferred by looking at the private unmasked bit value b_{i, σ_j} . Although initially used in a distributed scenario, we extend the general idea to the centralized scheme as well.

5.3.1. Assumption

Each user u_i generates a private random bit mask $s_i = [c_{i,1}, c_{i,2}, \dots, c_{i,m}]$, $c_{i,j} \in \{0, 1\}$, of the same length of the schedule x_i .

5.3.2. Scheme

The privacy-preserving scheduling algorithm is shown in Fig. 6. Each user first selects the sequence of time slots according to the permutation σ , i.e., $\sigma_j, \forall j=1, \dots, m$, and then masks the corresponding schedule bits, i.e. $b_{i, \sigma_j}^{\oplus} = b_{i, \sigma_j} \oplus c_{i,j}$. Then, each user encrypts individually both its bit mask, i.e. $E_i^c = [E_{K_p, r_{i,1}}(c_{i,1}), \dots, E_{K_p, r_{i,m}}(c_{i,m})]$, and the masked availabilities, i.e. $E_i = [E_{i, \sigma_1}, \dots, E_{i, \sigma_m}]$, where $E_{i, \sigma_j} = E_{K_p, r_{i,j}}(b_{i, \sigma_j}^{\oplus})$. Afterwards, each user u_i sends its E_i and E_i^c to the server, which computes the multiplication of the received E_{i, σ_j} with the encrypted masks of all other users $u_k, \forall k \neq i$, obtaining $E_{i, \sigma_j}^{\oplus} = E_{i, \sigma_j} \cdot \prod_{k \neq i} E_{K_p}(c_{k,j}), \forall i \in 1, \dots, N$ and $\forall j=1, \dots, m$. Afterwards, the server sends all individual schedules, masked by a global mask $c_{1,j} \oplus \dots \oplus c_{N,j}$, to each user in a random order. As a result, a user will not know his own schedule (masked with the global mask), otherwise he would be able to determine the global mask. Finally, each user decrypts the received messages and compares all masked individual schedules. If for a given time slot σ_j they all have the same value, then each user u_i can infer whether the time slot σ_j is available by looking at its own schedule b_{i, σ_j} . The following result shows the correctness and privacy properties of SchedGM.

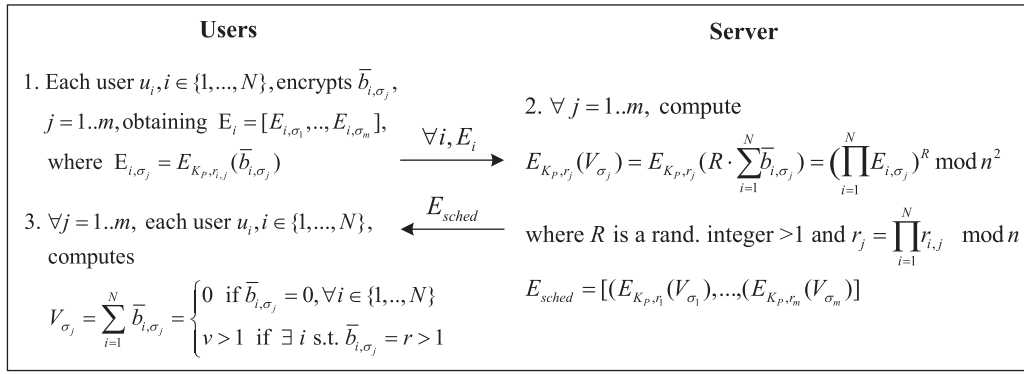


Fig. 5. SchedPa protocol.

Lemma 3. The protocol SchedGM is correct and server-private.

Proof. Correctness: From Section 3.3, we know that any scheduling algorithm should output $f(Y)$, on inputs f_1, f_2, \dots, f_N , where $f_i = f(b_{1,1}, \dots, b_{i,m})$, such that each user is able to privately compute $Y = f^{-1}(f(Y))$, where $Y = y^1, \dots, y^j, \dots, y^m$. The output bit y^j should be such that it should take (with overwhelming probability) some value v if and only if all users are available. Otherwise, the output bit y^j never takes value v and should take some other value, indicating that at least one user is not available. In the case of SchedGM, each $f(y^j)$ (output by the server) consists of N different bits, one for each user, where each bit is the corresponding $b_{i,j}$ (schedule bit j of user u_i) masked by a global mask. From Fig. 6, we can see that $y^j = \text{"YES"}$, for a particular user u_i , if and only if all of the N bits in $f(y^j)$ are equal and $b_{i,j} = 1$ (user u_i is available), and $y^j = \text{"NO"}$ otherwise. It is straightforward to see that all N bits in $f(y^j)$ will be equal only in two cases: 1) $b_{i,j} = 1, \forall i$ (all users are available) or 2) $b_{i,j} = 0, \forall i$ (all users are not available). Thus, $y^j = \text{"YES"}$ if and only if all users are available and $y^j = \text{"NO"}$ for any other case. Thus, SchedGM is correct.

Privacy: Hereafter we present the privacy proofs, both for user- and server-privacy.

• User identifiability advantage

As for the previous two algorithms, the identifiability advantage of any user u_i for the SchedGM protocol is defined as

$$Adv_{u_i}^{IDT}(SchedGM) = \left| Pr_{u_i}[b'_{j,p} = b_{j,p} | r > 1, b_{i,p}] - \frac{1}{2} \right|$$

where $1 \leq r \leq \lfloor N/2 \rfloor$ is the number of output elements that have the same value. Note that in SchedGM each user gets N masked output values $b_{i,p}^\oplus, \forall i \in \{1, \dots, N\}$, for each time slot $p \in \{1, \dots, m\}$, but it cannot unmask them as it does not possess the global mask. Therefore, any user knows that there are r masked bit values of one kind and $N - r$ of the other kind, without knowing whether one or the other kind corresponds to $b_{i,p} = 1$. Without

loss of generality, we assume that the Challenger chooses user u_1 as the Adversary and we focus on the non-trivial case $N > 2$. By expanding the first term, we have

$$\begin{aligned} & Pr_{u_1}[b'_{j,p} = b_{j,p} | r > 1, b_{1,p}] \\ &= \sum_{k=0}^1 Pr(b'_{j,p} = b_{j,p} | r > 1, b_{1,p} = k) \cdot Pr(b_{1,p} = k | r > 1) \\ &= \sum_{k=0}^1 \sum_{z=0}^1 Pr(b'_{j,p} = z | r > 1, b_{1,p} = k) \cdot Pr(b_{j,p} = z | r > 1, b_{1,p} = k) \cdot Pr(b_{1,p} = k | r > 1) \end{aligned}$$

From the above, we obtain

$$\begin{aligned} Pr(b'_{j,p} = 0 | r > 1, b_{1,p} = 0) &= \frac{1}{2} \cdot \frac{C_r^N \cdot r}{C_r^N \cdot N} + \frac{1}{2} \cdot \frac{C_{N-r}^N \cdot (N-r)}{C_r^N \cdot N} = \frac{1}{2} \\ Pr(b'_{j,p} = 1 | r > 1, b_{1,p} = 0) &= \frac{1}{2} \\ Pr(b'_{j,p} = 0 | r > 1, b_{1,p} = 1) &= Pr(b'_{j,p} = 1 | r > 1, b_{1,p} = 1) = \frac{1}{2} \end{aligned}$$

which implies

$$Pr_{u_i}[b'_{j,p} = b_{j,p} | r > 1, b_{i,p}] = \frac{1}{2}$$

and thus the final result

$$Adv_{u_i}^{IDT}(SchedGM) = 0, \quad \forall N > 2$$

• User linkability advantage

Hereafter we intuitively show that $\exists N > 2 | Adv_{u_i}^{LNK}(SchedGM) \geq 1/p(N)$, where $p(N)$ is any positive polynomial function of N . After Step 4 of the challenge-response game, the Adversary u_1 knows (i) its own schedule bit $b_{1,p}$ and (ii) the number r of masked schedules of one particular kind. Even though u_1 cannot determine

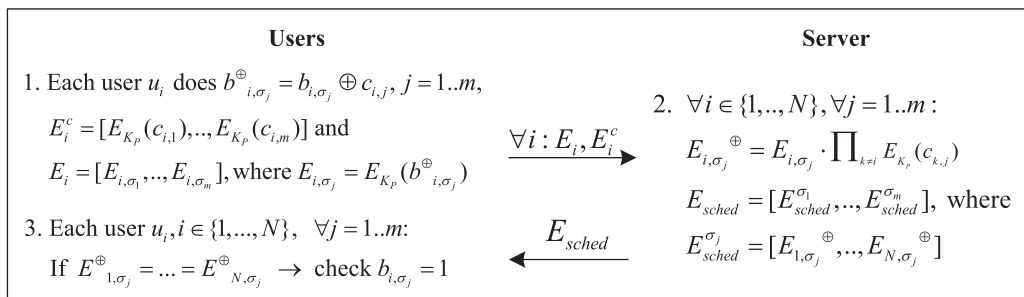


Fig. 6. SchedGM protocol.

Table 2
Client and server specifications.

	Client (Nokia N810)	Server
Processor	TI OMAP 2420 400 MHz	Intel Centrino Duo T2500, 2 × 2.00 GHz
RAM	DDR RAM 128 MB	DDR2 RAM 3 GB
OS	Maemo Linux OS2008 (Diablo)	Ubuntu 9.10, kernel 2.6.31.22

with certainty whether the r elements correspond to the “available” or to the “busy” state, it knows that the challenger picks the two other bits $b_{h,p}, b_{j,p}, j \neq h \neq i$, at random and therefore it also knows that the lower the value r , the greater the probability that any two bits in the sequence under consideration have the same value. Intuitively, if $r=1$ it means that there are $N-1$ schedules of one kind and only one schedule of the other kind. Therefore, the probability that any two users have same schedule value is greater than, for instance, when $r=\lfloor N/2 \rfloor$. Thus, the linkability advantage $Adv_{i_i}^{LNK}(SchedGM)$ is not less than $1/p(N)$, $\forall N > 2$, as $\exists r \in \{1, \dots, \lfloor N/2 \rfloor\} | Adv_{i_i}^{LNK}(SchedGM) \geq 1/p(N)$ for some positive polynomial $p(N)$.

- Server advantages

As in *SchedELG* and *SchedPa*, the server performing the *SchedGM* algorithm does not have access to any schedule bit. Therefore, *SchedGM* is server-private. \square

6. Implementation and performance evaluation

In this section we present the system and implementation details related to our three privacy-preserving scheduling algorithms. First, we describe the details about the systems and platforms on which we developed and implemented our applications. Second, we present the experimental measurements of the performance of our applications (both on the client devices and on the server), and we thoroughly discuss these results and compare the efficiency of all the algorithms.

6.1. Systems and platforms

Clients and server systems: The client application was run, tested and evaluated on the Nokia N810 devices. The server application was implemented and evaluated on a high-end machine. The hardware and OS specifications are listed in Table 2.

Code specifications: Our privacy-preserving scheduling applications were developed with the Qt 4.0 framework (Nokia Qt Framework, 2011), using QtCreator as the IDE. The client application was ported to the N810 devices using the Maemo SDK on the Scratchbox cross-compilation toolkit².

Cryptographic libraries: The *libgcrypt* standard GNU library³ was used to implement the ElGamal and the RSA cryptosystems. Similarly, the *libpaillier* library⁴ was used to implement the Paillier cryptosystem. For the Goldwasser–Micali cryptosystem, we did not find any existing available libraries, and therefore we developed a new library, *libgm*, to implement the basic cryptographic operations. We intend to release our *libgm* library to the public under the GPL licence.

² Details on the Scratchbox and Maemo SDK are available at <http://maemo.org/maemo.release.documentation/maemo4.1.x/node4.html>.

³ The documentation for *libgcrypt* is available at <http://www.gnupg.org/documentation/manuals/gcrypt/index.html>.

⁴ Source code available at <http://acsc.cs.utexas.edu/libpaillier/>.

6.2. Software architecture

Our privacy-preserving activity scheduling software consists of two applications: the *client* and the *server*. The client application runs on the Nokia N810 mobile device, and has a GUI to take inputs from the users. The server application runs on the Intel-based PC and is managed through the standard Unix console.

6.2.1. Client application

The client application stores the schedules of the users and displays the list of potential meeting participants for each user. This list is maintained and managed by the user himself, who can choose the meeting participants before initiating the meeting scheduling procedure. Each user can use the GUI to set his availabilities, send a meeting scheduling request, reply to an ongoing meeting request or refuse to participate in a received meeting request. To send a meeting scheduling request, the initiator first selects one of the available privacy-preserving algorithms (*SchedELG*, *SchedPa* or *SchedGM*) and the intended meeting participants. Then, the procedure is initiated by a click on the “Start meeting” button. Fig. 7a shows a flowchart of the application on the client device, when a user sends a request to schedule a meeting.

6.2.2. Server application

The server is a GUI-less application that interacts with the clients to handle requests such as login and computation of common availabilities. The main server class, *ScServer*, inherits *QTcpServer* and is used as the server socket. Fig. 8 shows the server flowchart structure.

More details about the inner structure of the server will be made available to the public, together with the source code, under the GPL licence.

6.3. Experimental performance evaluation

Before presenting the performance measurement details, let us first perform a comparative analysis of the asymptotic complexities of the proposed protocols, as shown in Table 3. In order to compare our three algorithms with an equivalent security, we set the bit-lengths of the ElGamal modulus q and the Paillier and GM modulus n to 1024 bits. A time-slot availability would then be encrypted to a 2-tuple of 1024-bit ciphertexts for ElGamal, to a 1024-bit ciphertext for GM and to a 2048-bit ciphertext for the Paillier encryption scheme.

From Table 3 we can see that the *SchedELG* and *SchedPa* protocols are very efficient, both in terms of communication $O(m)$, where m is the number of time slots, and computation complexity $O(m)$. Moreover, these two algorithms provide strong privacy guarantees. *SchedGM*, on the contrary, is comparatively less efficient due to the greater number of exchanged messages ($O(N \cdot m)$, where N is the number of participants). From the privacy perspective, *SchedGM* reveals more information: users can infer the ratio of free/busy participants for each time slot without identifying those that are busy and those that are free. Because in all schemes, the server operates only on encrypted data, it cannot gain any knowledge about the users’ private schedules.

Distributed (Silaghi and Mitra, 2004; Herlea et al., 2001) and hybrid (Yokoo et al., 2005) solutions proposed in the literature are less efficient from the communication standpoint as compared to the proposed protocols. Moreover, the computational complexity of these schemes is higher than *SchedELG* and *SchedPa*, and this undermines their applicability on resource-constrained mobile platforms. Even though the hybrid approach (Yokoo et al., 2005) has comparable computation complexity, it is not completely reliable from the privacy point of view because it assumes that the server(s) can get clear-text access to the individual availabilities.

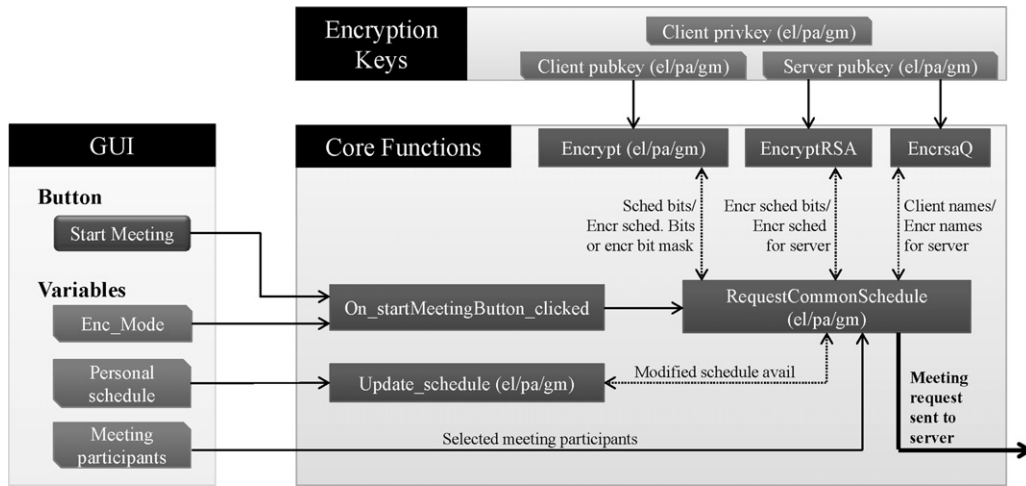


Fig. 7. Flowchart showing the initiation of a meeting scheduling request on the client application. The function names (such as encrypt (elg/pa/gm) or requestCommonSchedule (elg/pa/gm) that appear in this figure are intuitive placeholders for the actual function names that are used in the client application.

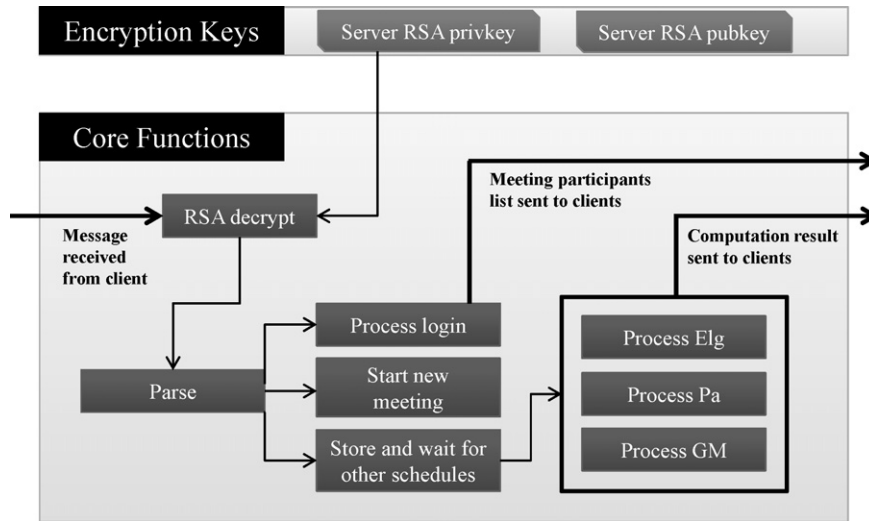


Fig. 8. Flowchart showing the server application structure when handling meeting requests and replies from and to the clients. The function names (such as RSA Decrypt or Process Elg) that appear in this figure are intuitive placeholders for the actual function names that are used in the server application.

We further evaluate the performance of *SchedElg*, *SchedPa* and *SchedGM* by implementing the client component of the protocols and primitives on Nokia N810 mobile devices with a 400 MHz CPU and 128 MB RAM (Fig. 9), and the server component on a desktop computer with a 2 GHz CPU and 3 GB RAM. The results of the experimentation are shown in Fig. 10.

6.3.1. Client encryption

As we can see from Fig. 10, the time required to perform the scheduling operations increases with the number of time slots for all the proposed algorithms, which is intuitive. With respect to encryption performance, Fig. 7a shows that *SchedElg* is the most efficient scheduling algorithm, requiring 4 s to encrypt 45 time slots (a typical weekly schedule on a per hour basis). The same task

Table 3 Efficiency and privacy comparison with the scheduling protocols DisCSP Yokoo et al., 2005, MPC-DisCSP2 Silaghi and Mitra, 2004 and SDC Herlea et al., 2001.

	Per-user encr.	Per-user decr.	Per-user comm.	Order of an encr. availab.	Privacy properties
Centralized					
SchedElG	$O(m)$	$O(m)$	$O(m)$	1024 bits	User-private Server-private
SchedPa	$O(m)$	$O(m)$	$O(m)$	2048 bits	User-private Server-private
SchedGM	$O(m)$	$O(N \cdot m)$	$O(N \cdot m)$	1024 bits	User-private ^b Server-private
Naïve	0	0	$O(m)$	1 bit ^a	None
Hybrid					
DisCSP protocol	$O(m)$	$O(m)$	$O(N \cdot m)$	1024 bits	Private
Distributed					
MPCDisCSP2 protocol	$O(N \cdot m)$	$O(m)$	$O(N \cdot m)$	2048 bits	Private
SDC protocol	$O(N^2 \cdot m)$	$O(N \cdot m)$	$O(N \cdot m \cdot \lceil \log_2(N) \rceil)$	1024 bits	Private

^a The naïve algorithm does not encrypt the schedule bits.

^b Adv^{DT} is a negligible function, whereas, for some output Y of the algorithm, Adv^{LNK} is non-negligible.

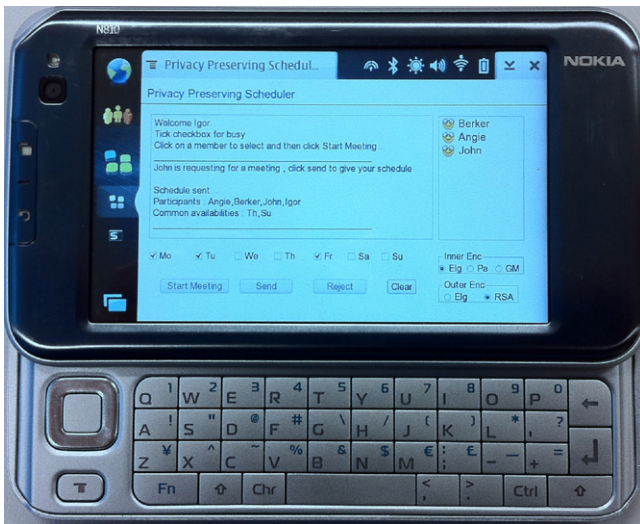
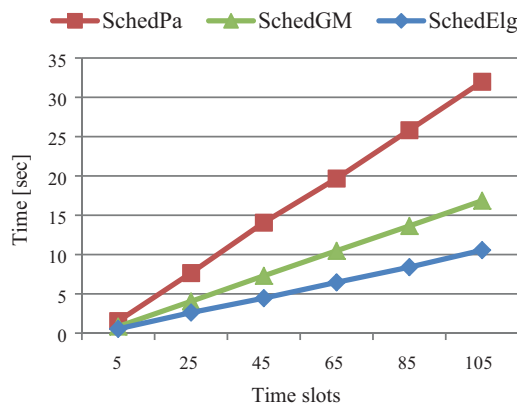


Fig. 9. Frontend of the scheduling application on a Nokia N810.

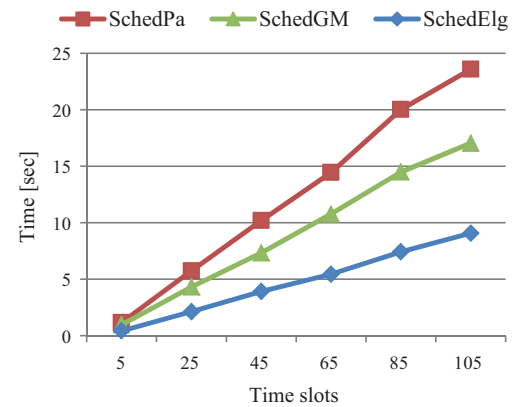
is accomplished by *SchedGM* and *SchedPa*, respectively, in 7 and 14 s. These results might be explained by the following. First, the cryptographic primitives for the ElGamal scheme are implemented in a standard well-optimized library, *libcrypt*, present in most Unix-based operating systems. *SchedGM*, on the contrary, does not use a standard library and can be further optimized. Second, the encrypted elements in *SchedPa* have twice the bit-length of those used in the other two algorithms, and therefore the same operations (multiplications and exponentiations) require more time.

6.3.2. Client decryption

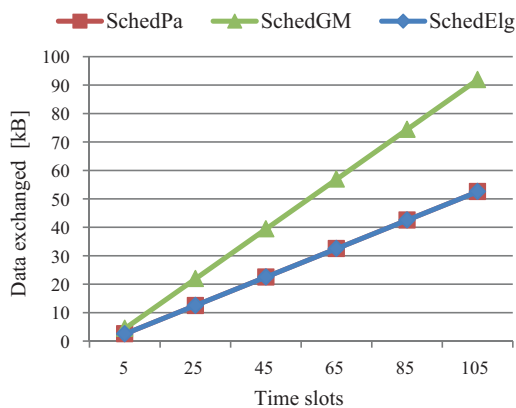
Fig. 7b shows the time required for decrypting the final result (common availabilities) of the scheduling algorithms at the client. Similarly to the encryption time, the fastest algorithm for the decryption is *SchedElg*, which takes 4 s in order to obtain the aggregated availabilities for a 45 time-slot period. For the same number of time slots, *SchedPa* takes approximately 7 s, which is almost twice longer than the best performance. The decryption times for both *SchedElg* and *SchedPa* are independent of the number of participants. The performance of *SchedGM*, due to the fact that the final output of the algorithm is a sequence of vectors instead of just a single aggregated vector, decreases



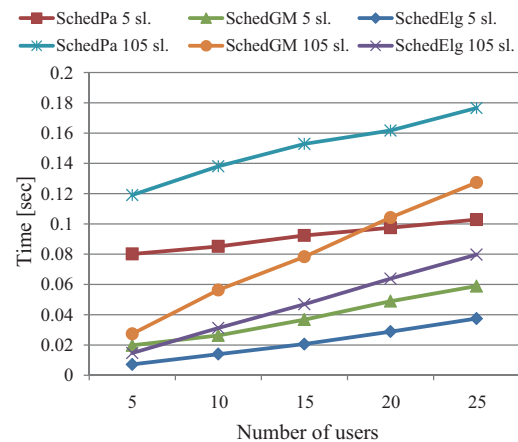
(a) Client encryption performance



(b) Client decryption performance. We considered $N = 5$ for *SchedGM*.



(c) Client communication efficiency (at application layer). We considered $N = 5$ for *SchedGM*.



(d) Server processing performance.

Fig. 10. Testbed implementation performance measurements.

with the number of users, as well as with the number of time slots. Thus, for a reasonable number of participants (e.g. $N=5$), *SchedGM* is still practical enough to be implemented on resource-constrained mobile devices, although it is not the preferred solution.

6.3.3. Client communication

Fig. 7c shows the (application layer) data that each client exchanges during one execution of the scheduling algorithm. In general, all the proposed privacy-preserving scheduling algorithms have reasonable communication costs. *SchedElg* and *SchedPa* are the most efficient algorithms and they require 22 kB of data in order to compute the aggregated availabilities of a 45 time-slot period, whereas *SchedGM* requires 39 kB for the same result. As previously mentioned, *SchedGM* uses a sequence of masked vectors in order to compute the final availabilities of the users, and therefore the amount of data is proportional both to the number of users and time-slots.

6.3.4. Server performance

The scheduling server's performance is shown in Fig. 8. As it can be seen, the time required to perform the scheduling operations on encrypted values increases with both the number of users and time slots. For instance, the running time (in seconds) for the server implementation of the *SchedElG* algorithm is at most $2 \cdot N \cdot m \cdot T_{mul-ElG}$, where N is the number of clients, m is the number of time-slots and $t_{mul-ElG}$ is the time required to compute one multiplication operation between two $\lceil \log(q) \rceil$ -bit integers (q is the order of the group in the ElGamal encryption scheme). The running time for the *SchedPa* and *SchedGM* is, respectively, at most $N \cdot m \cdot T_{mul-Pai} + m \cdot T_{exp-Pai}$ (where $T_{mul-Pai}$ and $T_{exp-Pai}$ is the time required to perform a multiplication and an exponentiation respectively of two $\lceil \log(n^2) \rceil$ -bit integers) and $3 \cdot N \cdot m \cdot T_{mul-GM}$ (where T_{mul-GM} is the time required to perform a multiplication between two $\lceil \log(n) \rceil$ -bit integers).

As it can be seen, even with a large number of users and time slots, the amount of time required for the server-side scheduling operations is still below 0.2 s, which suggests that the load on the server is limited, which allows it to efficiently handle multiple scheduling events, without incurring in huge computational overhead.

7. Extensions

In this section, we show how *SchedPa* can be easily extended to the case where user schedules are non-binary, i.e., each time slot is a non-negative cost $C_{i,j}$ that indicates u_i 's preference for time-slot j . We also describe several active attacks on the proposed scheduling schemes, such as collusion between users-server and data modifications by the users, and how these attacks can be mitigated using existing cryptographic mechanisms. Finally, we discuss some further enhancements for the privacy of users' schedules and how to implement them.

7.1. Non-binary schedules

The goal here is to find, in a privacy-preserving fashion, the time-slot with the minimum aggregated cost. The scheme works as follows:

1. Each user u_i reorders his cost sequence $C_{i,1} \dots C_{i,m}$ using the shared permutation σ and encrypts each cost C_{i,σ_j} in the sequence using the Paillier cryptosystem with the shared group

key K_p . He then passes the result $(E_{K_p,r_{i,1}}(C_{i,\sigma_1}) \dots E_{K_p,r_{i,m}}(C_{i,\sigma_m}))$ to the server.

2. The server computes the encrypted sum of costs $E_{K_p,r_j}(R \cdot \sum_{i=1}^N C_{i,\sigma_j})$, $\forall j$, where R is a random integer (greater than one) chosen by the server.
3. The server selects a pre-determined user u_k and passes a *randomly ordered* (different from σ) sequence of the encrypted aggregated costs to it. This is to prevent u_k from learning the aggregated cost function.
4. User u_k decrypts all the elements passed from the server, and identifies the minimum aggregated cost.
5. User u_k then queries the server for the index of the (encrypted) minimum aggregated cost. The server then distributes the queried index to all users.

It can be easily shown that the above scheme is execution privacy-preserving. For conciseness, we do not discuss the details of the privacy analysis here.

7.2. Active attacks

There are five kinds of possible active attacks on the scheduling schemes: (i) collusion between the scheduling server and users, (ii) collusion among users, (iii) fake user generation by the server, (iv) individual user schedule modification and (v) integrity and replay attacks.

In order to thwart the first issue, the invited participants could agree to establish a shared secret using techniques from threshold cryptography, such as Stadler (1996). The server should then collude with at least a predefined number of participants in order to obtain the shared secret and learn the individual availabilities. The second concern may arise if k colluding users set their schedules to *all-available*, and try to learn the schedules of other users. Assuming that N is the total number of participants and k the number of colluding ones, our schemes would provide some level of schedule privacy to honest users, as long as $N - k \geq 2$. Only if all but one users collude, then they would be able to determine the schedule of the remaining user. In order for the third attack to succeed, the server would need to generate fake users and convince the true participants about the legitimacy of the fake users. In practice, this is a non-trivial task to achieve, and thus the attack has a very slim chance of succeeding. Moreover, the effectiveness of such an attack could be further reduced by adopting the threshold cryptographic scheme mentioned previously, because the server would then need to generate k fake users and validate them as true participants.

The fourth attack is also not able to succeed in revealing the availability of other meeting participants, as the best a malicious user can do is to set its own schedule to *all-available*, and then guess the availabilities of the other $N - 1$ participants. Even if a malicious user attempts to modify its own schedule with invalid values, such as negative values, the message domain restrictions of cryptosystems (such as ElGamal and Paillier) would prevent such modifications. Thus, malicious attacks consisting of manipulating the final result using invalid negative values as schedule values are not possible in the proposed protocols.

The last attack concerns the integrity and freshness of the encrypted schedules. The participants are the only entities in the system that know the secret that has been used to generate the public/private key pair, and therefore they are the only ones that can generate and verify the integrity of the encrypted data. Moreover, using the shared common secret, each participant could generate a fresh *nonce* at each algorithm execution and send it (in encrypted form) to the server during the scheduling process. The server would then forward these encrypted nonces to each participant, who

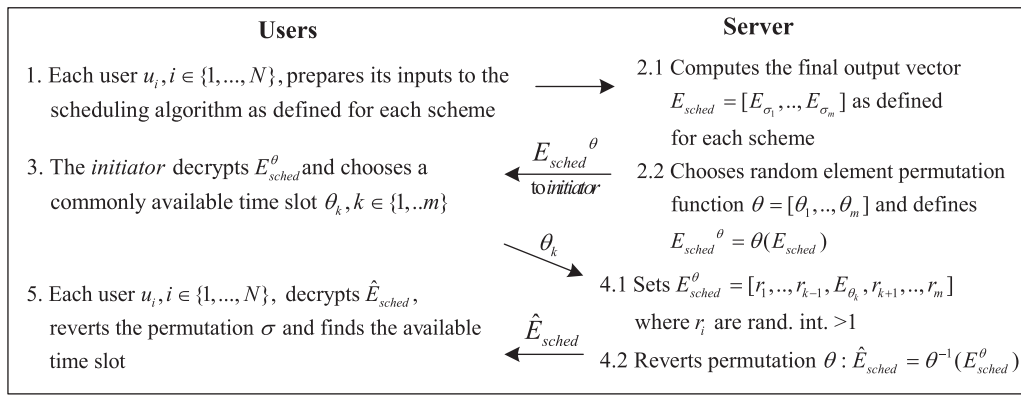


Fig. 11. Extended algorithm scheme for revealing a single available time slot.

could verify that all received nonces are equal. If not all nonces are equal, then the participants know that there has been at least one replay attack, and thus the schedule results are not to be trusted.

7.3. Single available time slot

The output of conventional, no-privacy-preserving scheduling services (such as Doodle or Outlook) consists of time slots in which all participating users are available. The proposed schemes follow this paradigm and they provide, in an efficient and privacy-preserving way, all time-slots for which all users are available.

In some cases, however, it might be desirable to limit the disclosure of common availabilities to only one time-slot, instead of the set of all available time-slots. This would provide an additional layer of privacy for the individual schedules, as the participants would be given a single feasible solution. Hereafter we describe one simple way to adapt the proposed schemes to support this feature (Fig. 11).

First, all users participating in the scheduling process perform Step 1 of the respective algorithm (SchedElg, SchedPa or SchedGM). Second, the server performs Step 2 but it does not send the final output to each user. Instead, it randomly chooses a private time-slot permutation function $\theta = [\theta_1, \dots, \theta_m]$ and applies it to the elements of the final output vector(s) E_{sched} . We call this new vector(s) E_{sched}^θ . At this point, the schedules have been permuted twice, once by the users prior to the encryptions (with σ) and once by the server (with θ).

Next, the server sends E_{sched}^θ to the user who started the activity scheduling (the *initiator*), which then gets the common availabilities but in a doubly permuted order. The initiator is able to determine the *available* slots in this doubly permuted time slot list, but he is not able to determine the time slots they correspond to in the original schedule. The initiator selects one commonly available time slot θ_k and securely sends the index θ_k to the server. Fourth, the server (i) replaces all availabilities other than θ_k in E_{sched}^θ with random numbers, (ii) reverts the permutation θ , and (iii) sends this new vector(s) \hat{E}_{sched} to each user. Finally, each user decrypts and reverts the initial permutation σ of the received vector(s) and determines which time slot j is the only commonly available time slot.

This simple solution that reveals only a single available time slot to all the participants involves one extra message exchange between the initiator and the scheduling server, as shown in Step 3 of Fig. 11. Although the permutation θ performed by the server prevents the initiator from knowing the true common availabilities, he might still want to maliciously modify the permuted availabilities. However, the only action the initiator can do is to choose one of the permuted time slots and communicate its index θ_k to the

server, as it is the server who will then revert the permutation θ and send the final vector(s) \hat{E}_{sched} to all users.

8. User study

In this section we present the modalities and results of the user study that we carried out with our prototype meeting-scheduling application. The goal of this study was to assess the sensitivity of the subjects to privacy issues in meeting-scheduling applications, as well as to obtain feedback with respect to our prototype application.

8.1. Background

Based on the privacy- and usability-related questionnaire guidelines from (Chignell et al., 2003; Lewis, 1995), we prepared and conducted a targeted user-study on 19 subjects, sampling a population of university students (both undergraduate and graduate), non-scientific personnel and people from a non-technical environment.

The entire study was divided into three phases, with two different sets of questions that were given in Phase 1 and Phase 3 respectively. In Phase 1, the participants were asked to reply to a set of 20 questions before using the meeting scheduling application. In Phase 2, they were asked to use our prototype application to schedule meetings with the other participants both in a controlled and uncontrolled setting; the first time, we instructed them how to use the application, and afterwards they were free to use it as they pleased. Finally, in Phase 3 the participants answered a second set of 14 post-experience questions, after having used our prototype application.

The goal of Phase 1 of the study was to assess the participants' level of adoption of mobile technology and applications, and to get their opinion on privacy issues in such applications. The participants were not told beforehand what kind of mobile application they will be asked to use in Phase 2. During Phase 1, the respondents answered the *Pre-Experience A* questionnaire, which comprises 20 questions on both generic technology topics (such as usage and ownership of mobile devices, utilization of mobile social networks and calendar/agenda) and more specific privacy-related questions (such as their online behavior and opinions on information release). For instance, one statement related to users' online behavior and privacy is "I am willing to use my real name in online discussions (forums, chat rooms, etc.)", to which the respondents had to answer with either *Disagree*, *Tend to disagree*, *Tend to agree* or *Agree*.

After Phase 1 was completed, in Phase 2 we instructed the participants on the specifics of our prototype scheduling application and how it works, in a step-by-step fashion. We then asked them to

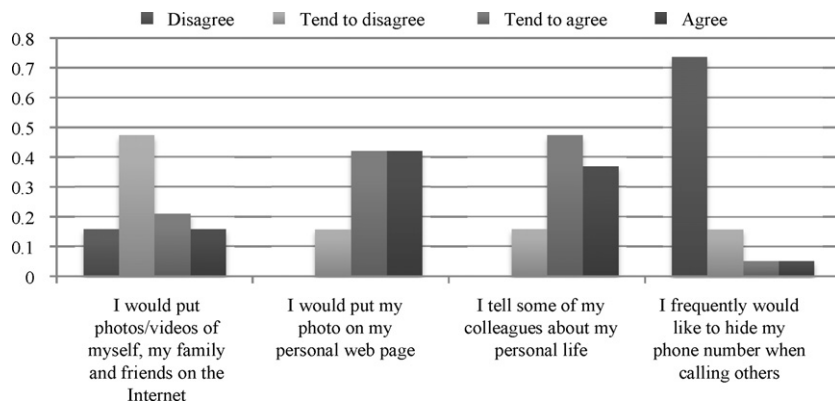


Fig. 12. Extract of the user-study questionnaire about people's privacy attitudes.

execute one instance of the scheduling process. Next, we told them to use the application as they please, without the experimenters overseeing the process. The goal of Phase 2 was to show our application and to let the participants use it autonomously, in order to get an opinion for the Phase 3 of the study.

The goal of Phase 3 was to obtain feedback on different performance and privacy aspects of our prototype application. The respondents answered the *Post-Experience B* questionnaire, which comprises 14 questions centered on our application prototype, its perceived usefulness, efficiency, ease of use, and privacy. For example, the statement "I could easily identify who was/were the person/people that were not available for a particular time slot" could be answered by *Disagree*, *Tend to disagree*, *Tend to agree* or *Agree*.

Hereafter we provide the summary of the results and discussion on our user-study.

8.2. Results

8.2.1. Phase 1

Technology utilization: In this first part, we discuss the results concerning the technology utilization habits of the respondents. With respect to mobile applications, our results show that 63% of the respondents browse the Internet with a mobile device, whereas 53% of them use the mobile calendar/agenda application on their devices in order to organize meetings. 86% of such meetings are scheduled once or twice a week, and most of the time (89%) such meetings involve 2–4 people. In order to reach a consensus, the meeting participants use e-mail 58% of the time and the telephone for the remaining 42%. Social networks, such as Facebook or Twitter, are used by 84% of the respondents, and 44% of them access such services using their mobile devices. These results suggest that although meeting scheduling and calendar management using mobile devices is already a reality, people still struggle to reach a consensus in an efficient way. In order to agree on a common time slot using e-mail, multiple rounds of interaction among the meeting participants are required.

Privacy attitudes: In this second part, we discuss the privacy concerns of the respondents when using everyday applications. In general, 63% of the respondents tend to disagree or disagree with the statement "I would put photos/videos of myself, my family and friends on the Internet". When asked about third parties sharing personal information about them, 89% of the respondents agree that no third party should disseminate users' private information without their knowledge. With respect to privacy in online interactions, 63% feel that they would prefer not to use their real name and use pseudonyms instead. Fig. 12 shows other interesting privacy attitude results. In summary, our respondents tend to be sensi-

tive to the privacy issues related to the use of mobile applications, and thus effectively controlling the access to and dissemination of personal information is a valuable differentiator for mobile applications.

Scheduling applications and privacy: The third part of the results show the opinion of respondents about meeting-scheduling applications on mobile devices. According to the results, 84% of the respondents are not aware of any existing mobile application for meeting scheduling. Among those, 43% would be quite (or a lot) interested in having such applications. With respect to privacy, 58% would be comfortable in sharing their basic schedule availabilities with the other meeting participants, while none of them would be willing to share all the details (such as place, time and subject) about these availabilities.

With respect to priorities in mobile meeting-scheduling applications, Fig. 13 shows the choices of the respondents, ordered by the perceived priority (on a scale from 1 to 4, where 1 is the top priority and 4 is the least priority). The figure shows that privacy is perceived as the first priority in mobile meeting-scheduling applications 33% of the time. If we consider the cumulative result for the 1st and 2nd priorities, privacy achieves a total of 77%. Although the ease of use of the application is perceived as the top priority for 50% of the respondents, the cumulative result for the 1st and 2nd priorities achieves 67%, which is 10% less than privacy. The speed and the Graphical User Interface (GUI) have the least priority for the users, where speed is only the third priority most of the time, and the GUI is almost exclusively the least priority.

Overall, the results suggest that privacy is indeed perceived as being the top or the second priority in meeting-scheduling applications, which is in line with the concerns that the respondents had before using our application. From a software developer standpoint,

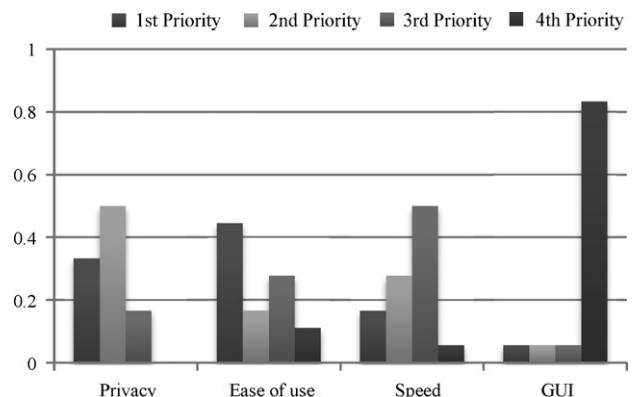


Fig. 13. Extract of the user-study questionnaire about people's priorities in mobile scheduling applications.

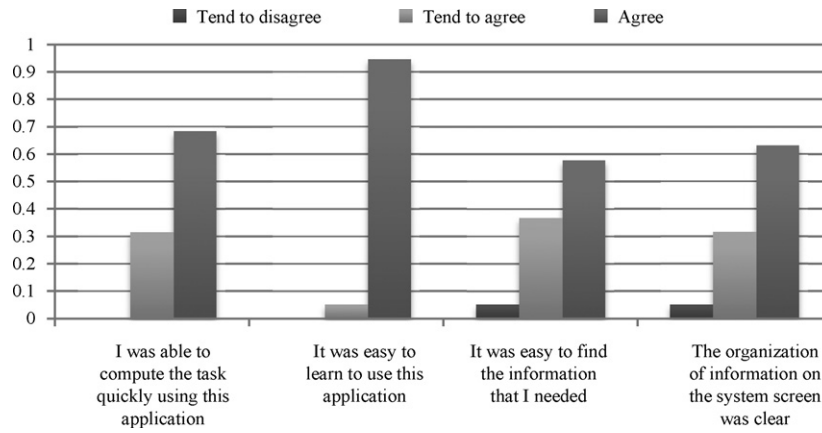


Fig. 14. Extract of the user-study questionnaire about the user experience for our prototype application.

this means that both ease of use and privacy need to be taken into account from the beginning of the application development process. In particular, the privacy mechanisms should be implemented in a way that does not significantly affect the usability or performance. The acceptance of meeting-scheduling applications is thus highly influenced by the availability of effective and intuitive means for controlling privacy preferences.

8.2.2. Phase 3

User-experience: Fig. 14 shows some interesting results about the perceived user experience while using our prototype meeting-scheduling application on the Nokia N810 devices. As it can be seen, almost 70% of the respondents agree that they were able to perform the meeting scheduling task quickly using our application. Moreover, 95% of them agree that it was easy for them to learn to use our application. Regarding the information presented on the screen, users mostly agree that it was easy to find all necessary information, such as the meeting participants, the individual schedule and control buttons. Similar results have been obtained for the organization of the user interface.

These results suggest that it is indeed possible to integrate simple privacy mechanisms into mobile application, without incurring in significant learning overhead. A clean GUI with a transparent integration of privacy features proved to be very effective in this regard.

Privacy in our prototype application: In this last part, we discuss the subject of privacy with respect to our prototype application, and how its implementation was perceived by the respondents. Fig. 15 shows some of the results obtained from the user study. In general, all respondents tend to agree that it is important to not reveal any more information to the central server than strictly necessary. When asked about the way privacy has been implemented in our prototype application, 95% of them claim that they could not identify the people who were available (or not) in a given time-slot. Concerning the potential overhead due to the privacy mechanisms, 71% of the users feel that having the privacy feature in such application did not make it more complicated for them to use it; only 5% tend to agree with the opposite.

Regarding the third-party knowledge of the individual schedules, 74% agree that they felt comfortable knowing that the central scheduling server did not know their private schedules, and only 5% of them disagree. The users were told about this feature during Phase 2 of the study. However, when the third-party is the other meeting participants (and not the central server), 47% felt comfortable knowing that their privacy was preserved. Nevertheless, this percentage increases to 95% when considering responders who tend to agree with such statement, in addition to those who agree.

In summary, this user-study has shown that the majority of the respondents are concerned about their privacy in scheduling applications, and that they would welcome effective and simple means for protecting it and still enjoy such services. Our prototype appli-

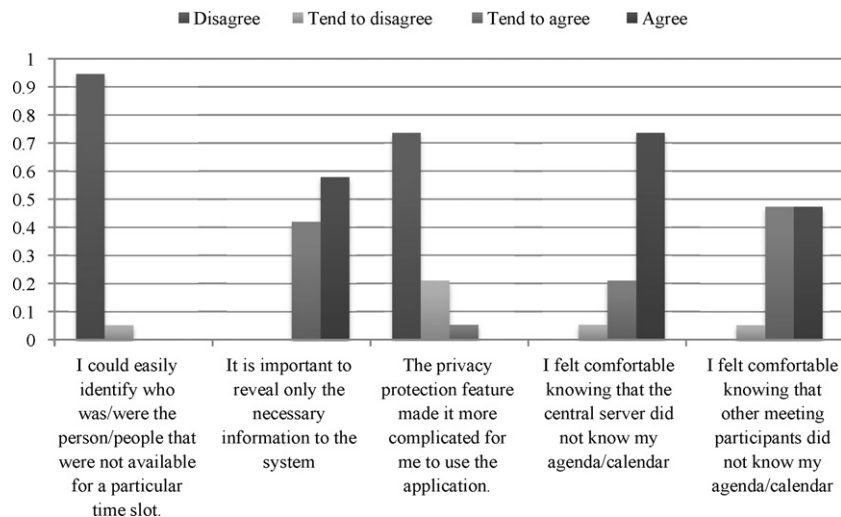


Fig. 15. Extract of the user-study questionnaire about people's opinions on the privacy features in our prototype application.

cation has proven to be effective in both providing a user-friendly interface for the meeting scheduling participants, and a transparent way to ensure that privacy of individual schedules is preserved. The results have also shown that there is no significant overhead for using privacy in such applications, and that people appreciated having the ability to not disclose more information about their schedules than what was strictly necessary in order to compute the available time slots.

9. Conclusion and future work

Activity-scheduling applications are increasingly used by people on-the-move in order to efficiently and securely manage their time. In addition to privacy, which is paramount, such services should also be practical and feasible to implement, given the client-server paradigm that most providers use, and they should be as transparent to the user as possible. In this paper, we have provided a framework for the formal study of privacy properties in such applications, and we have proposed three novel privacy-preserving protocols that, in addition to guaranteeing privacy, are more efficient than similar solutions in terms of computation and communication complexities. Our implementation and extensive performance evaluation on real mobile devices demonstrated that the proposed privacy-preserving schemes are well suited to practical network architectures and services. Moreover, a thorough user-study of the prototype application suggests that our algorithms and software architecture are seamlessly integrated with the privacy-preserving algorithms, in a way that does not impede the user from quickly and effectively utilizing our application.

As part of our future work, we intend to further optimize the implementation of the proposed scheduling algorithms for performance on mobile devices, and to include user preferences and the security related features described in Section 7. We also plan to release the source code of the proposed scheduling schemes to the general public under the GPL licence.

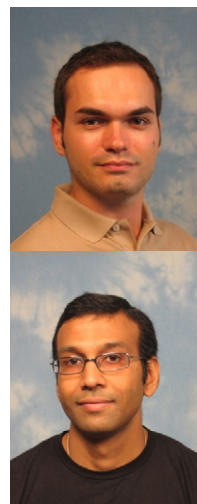
Acknowledgments

We would like to thank Mathias Humbert, Nevena Vratonjic, Anthony Durussel and Gianpaolo Perrucci for their constructive input that helped improving the quality of this work, as well as the Nokia Research Center (Lausanne) for supporting this project.

References

- Apple iCal. <http://apple.com/ical> (Last visited 27.01.2011).
- Bellare, M., 2008. A note on negligible functions. *Journal of Cryptology* 15, 271–284, doi:10.1007/s00145-002-0116-x.
- Chacin, C., Strobl, R., 2004. Asynchronous group key exchange with failures. In: PODC '04: Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing, ACM, New York, NY, USA, pp. 357–366.
- Chen, C.H.O., Chen, C.W., Kuo, C., Lai, Y.H., McCune, J.M., Studer, A., Perrig, A., Yang, B.Y., Wu, T.C., 2008. Gangs: gather, authenticate 'n group securely. In: MobiCom '08: Proceedings of the 14th ACM International Conference on Mobile Computing and Networking, ACM, New York, NY, USA, pp. 92–103.
- Chignell, M., Quan-Haase, A., Gwizdka, J., 2003. The privacy attitudes questionnaire (paq): initial development and validation. In: Human Factors and Ergonomics Society Annual Meeting Proceedings.
- CHILabs PDA (Personal Digital Assistants) Use Study. <http://personal.bgsu.edu/~nberg/chilabs/pda.htm> (Last visited 27.01.2011).
- Dailywireless.org. <http://www.dailywireless.org/2009/03/24/smartphone-users-100m-by-2013> (Last visited 27.01.2011).
- De Cristofaro, E., Tsudik, G., 2010. Practical private set intersection protocols with linear complexity. In: Financial Cryptography and Data Security FC'10.
- Doodle: Easy Scheduling. <http://www.doodle.com/> (Last visited 27.01.2011).
- Du, W., Atallah, M., 2001. Secure multi-party computation problems and their applications: a review and open problems. In: Proceedings of the 2001 Workshop on New Security Paradigms, ACM, New York, NY, USA, pp. 13–22.
- ElGamal, T., 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 31, 469–472.

- Ephrati, E., Zlotkin, G., Rosenschein, J.S., 1994. Meet your destiny: a non-manipulable meeting scheduler. In: CSCW '94: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, ACM, New York, NY, USA, pp. 359–371.
- Franzin, M., Freuder, E., Rossi, F., Wallace, R., 2004. Multi-agent meeting scheduling with preferences: efficiency, privacy loss, and solution quality. *Computational Intelligence* 20.
- Goldreich, O., 2001. Foundations of Cryptography, vol. 1. Cambridge University Press.
- Goldwasser, S., Micali, S., 1984. Probabilistic encryption. *JCSS* 28, 270–299.
- Google Smart Rescheduler. <http://gmailblog.blogspot.com/2010/03/smart-rescheduler-in-google-calendar.html> (Last visited 27.01.2011).
- Herlea, T., Claessens, J., Preneel, B., Neven, G., Piessens, F., De Decker, B., 2001. On securely scheduling a meeting. In: Trusted Information: The New Decade Challenge: IFIP TC11 16th International Conference on Information Security (IFIP/Sec'01), Paris, France, June 11–13, 2001. Kluwer Academic Pub., pp. 183–198.
- Kellermann, B., Böhme, R., 2009. Privacy-enhanced event scheduling. In: IEEE International Conference on Computational Science and Engineering, pp. 52–59.
- Kissner, L., Song, D., 2005. Privacy-preserving set operations. *Advances in Cryptology – CRYPTO 2005*, 241–257.
- Lewis, J., 1995. IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction* 7.
- Lin, Y.H., Studer, A., Hsiao, H.C., McCune, J.M., Wang, K.H., Krohn, M., Lin, P.L., Perrig, A., Sun, H.M., Yang, B.Y., 2009. Spate: small-group PKI-less authenticated trust establishment. In: MobiSys '09: Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services, ACM, New York, NY, USA, pp. 1–14.
- Microsoft Outlook. <http://office.microsoft.com/outlook> (Last visited 27.01.2011).
- Nokia Ovi. <http://ovi.nokia.com> (Last visited 27.01.2011).
- Nokia Qt Framework. <http://qt.nokia.com/> (Last visited 27.01.2011).
- Paillier, P., 1999. Public-key cryptosystems based on composite degree residuosity classes. *Advances in Cryptology – EUROCRYPT '99* 1592, 223–238.
- Rivest, R., Shamir, A., Adleman, L., 1978. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21, 126.
- Silaghi, M., Mitra, D., 2004. Distributed constraint satisfaction and optimization with privacy enforcement. In: 3rd IC on Intelligent Agent Technology, pp. 531–535.
- Silaghi, M.C., 2004. Meeting scheduling guaranteeing n/2-privacy and resistant to statistical analysis (applicable to any discsp). In: WI '04: Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence, IEEE Computer Society, Washington, DC, USA, pp. 711–715.
- Stadler, M., 1996. Publicly verifiable secret sharing. In: *Advances in Cryptology – EUROCRYPT '96*, pp. 190–199.
- Wallace, R., Freuder, E., 2005. Constraint-based reasoning and privacy/efficiency tradeoffs in multi-agent problem solving. *Artificial Intelligence* 161, 209–227.
- Yokoo, M., Suzuki, K., Hirayama, K., 2005. Secure distributed constraint satisfaction: reaching agreement without revealing private information. *Artificial Intelligence* 161, 229–245.
- Zunino, A., Campo, M., 2009. Chronos: a multi-agent system for distributed automatic meeting scheduling. *Expert Systems with Applications* 36, 7011–7018.



Igor Bilogrevic is a research assistant and PhD student at the Laboratory for computer Communications and Applications (LCA1), at the Ecole Polytechnique Fédérale de Lausanne (EPFL), under the supervision of Prof. Jean-Pierre Hubaux. He earned his M.Sc. and B.Sc. in Communication Systems from EPFL in 2009 and 2007 respectively, with specialization in Wireless Communications. Igor is a member of the ACM and IEEE. His main areas of interest include wireless networks and, in particular, security and privacy issues thereof. <http://people.epfl.ch/igor.bilogrevic>.

Murtuza Jadhliwala is a senior researcher with the Laboratory for computer Communications and Applications (LCA1) group, led by Prof. Jean-Pierre Hubaux, since September 2008. He completed his Doctoral (Ph.D.) degree in Computer Science from the Computer Science and Engineering Department at the The University at Buffalo, State University of New York in September 2008. He was a Ph.D. Candidate at the Computer Science and Engineering Department at the University at Buffalo since August 2006. He earned his Masters of Science (M.S.) degree in Computer Science from the University at Buffalo, State University of New York in June 2004, and his Bachelors degree (B.E.) in Computer Engineering from Fr. Conceicao Rodrigues College of Engineering, Mumbai University in June 2000. He worked as a Technical Leader at the Investment Research and Information Services Ltd., Mumbai, India from June 2000 until June 2002. Murtuza's research interests are in the area of Secure and Robust Communication-based Services (e.g., Localization and Time Synchronization) in Wireless Data Networks (Wireless LANs, Mobile Ad-hoc and Sensor Networks), System and Network Security, Privacy, Vulnerability Analysis, Combinatorial Optimization, Approximation Algorithms and Multi-Agent Systems including game theory and economic mechanism design. He is a member of the IEEE and the ACM. <http://people.epfl.ch/murtuza.jadhliwala>.



Praveen Kumar is an undergraduate student in the Department of Computer Science and Engineering at Indian Institute of Technology, Kharagpur. He is currently pursuing Bachelors of Technology degree course at IIT Kharagpur (2007–2011). In 2010, Praveen was a summer intern at LCA1, EcolePolytechniqueFederale de Lausanne, Switzerland, where he contributed to the development, analysis and testing of the application prototype for Privacy Preserving Scheduler. In 2009, he worked on a research project in networks and developed efficient routing schemes for scale-free networks. His main research interests are in the area of computer networks, cryptography and algorithms.



Sudeep is a fourth year undergraduate student in Computer Science Engineering, at Indian Institute of Technology, Kharagpur, India. He is enrolled in Bachelors+Masters of Technology at IIT Kharagpur(2007–2012). In 2010, Sudeep was a summer Intern at LCA1, EcolePolytechniqueFederale de Lausanne, Switzerland where he contributed to the development, analysis, and testing of the application prototype for Privacy Preserving Scheduler. In 2009, Sudeep was a summer Intern at Telecom and Management, Sudparis, France where he was involved in development of “PLUGS: Secrets of Museums”, a pervasive game. His main areas of interest include Cryptography, Database

Management systems and Networks.



Jean-Pierre Hubaux joined the faculty of EPFL in 1990. His research activity is focused on wireless networks, with a special interest in security and cooperation issues. In 1991, he designed the first curriculum in Communication Systems at EPFL. He was promoted to full professor in 1996. In 1999, he defined some of the main ideas of the National Competence Center in Research named “Mobile Information and Communication Systems” (NCCR/MICS). In this framework, he has notably defined, in close collaboration with his students, novel schemes for the security and cooperation in wireless networks; in particular, he has devised new techniques for key management, secure positioning, and incentives for cooperation in such networks.

In 2003, he identified the security of vehicular networks as one of the main research challenges for real-world mobile ad hoc networks. In 2008, he completed a graduate textbook entitled “Security and Cooperation in Wireless Networks”, with Levente

Buttayan. Most of his current research activities revolve around privacy issues in mobile networks and are partially funded by Nokia. He is co-founder and chairman of the steering committee of WiSec (the ACM Conference for Wireless Network Security). He has served on the program committees of numerous conferences and workshops, including SIGCOMM, INFOCOM, MobiCom, MobiHoc, SenSys, WiSe, and VANET. Since 2007, he has been one of the seven commissioners of the Federal Communications Commission (ComCom), the “Swiss FCC”. He held visiting positions at the IBM T.J. Watson Research Center and at UC Berkeley. He has been on the advisory board of Deutsche Telekom Laboratories (T-Labs) since their creation in 2004. He is a Fellow of both IEEE and ACM. He was born in Belgium, but spent most of his childhood and youth in Northern Italy. After completing his studies in electrical engineering at Politecnico di Milano, he worked 10 years in France with Alcatel, primarily in the area of switching systems architecture and software. <http://people.epfl.ch/jean-pierre.hubaux>.



Imad Aad is a Principal Researcher at Nokia Research Center (NRC) in Switzerland, working on privacy in contextual services. Before joining NRC in 2009, he was a senior researcher at DoCoMo Euro-Labs, Germany, working within the Future Networking Lab on cooperative coding, anonymity, quality of service, beam antennas and overlays in wireless networks. He got his Electrical and Electronics engineering degree in 1998 from the Lebanese University, Beirut. He got his M.S.degree in 1999 from the University of Nice - Sophia Antipolis, then his Ph.D. degree from Joseph Fourier University, France, in 2003. He prepared his Ph.D. on quality of service in wireless LANs at INRIA, France, within the Planète team. He worked at EPFL, Switzerland, within the LCA team as a senior researcher from 2003 to 2005 where he worked on cheating and cheating detection issues, DoS attacks and resilience and on general security aspects in wireless networks. <http://imad.aad.name>.

Switzerland, within the LCA team as a senior researcher from 2003 to 2005 where he worked on cheating and cheating detection issues, DoS attacks and resilience and on general security aspects in wireless networks. <http://imad.aad.name>.



Valtteri Niemi received a PhD in Mathematics from the University of Turku, Finland in 1989. He was an Associate Professor at the University of Vaasa, Finland, during 1993–97 after which he joined Nokia Research Center (NRC). Dr.Niemi has contributed in several roles for Nokia wireless security research. He was nominated as a Nokia Fellow in 2009 and currently leads security and networking team in the NRC Radio Systems laboratory. Dr.Niemi was the chairman of 3GPP security standardization group during 2003–2009. He has (co-)authored four books, more than 40 scientific articles and around 20 patents.