Open Research Questions of Privacy-Enhanced Event Scheduling

Benjamin Kellermann

Technische Universität Dresden
Institute of Systems Architecture
D-01062 Dresden, Germany
Benjamin.Kellermann@tu-dresden.de

Abstract. Event-scheduling applications like Doodle have the problem of privacy relevant information leakage. A simple idea to prevent this would be to use an e-voting scheme instead.

However, this solution is not sufficient as we will show within this paper. Additionally we come up with requirements and several research questions related to privacy-enhanced event scheduling. These address privacy, security as well as usability of privacy-enhanced event scheduling.

Keywords: event scheduling, electronic voting, superposed sending, anonymity, privacy-enhanced application design.

1 Introduction

For years, technical event scheduling has been done with applications like Microsoft Exchange or a central iCal-server within some intranet. Since Web 2.0 applications became very popular, an application named Doodle [1] which assists event scheduling in a very easy way became popular as well. In this solution, an initiator configures an online poll where everybody votes on dates that are shown to all visitors of the web page.

In contrast to the intranet solution, Doodle offers its event scheduling service to everyone. However, both solutions – the intranet solution and Doodle – have in common that everybody has to publish his so-called *availability pattern*. In contrast to a solution running in a company, within the Doodle-solution, the availability patterns are visible to the whole world.

An availability pattern contains at least two different types of information. The *direct inference* from the pattern are information which one can extract from the availability of certain dates ('Will my husband vote for the date of our wedding anniversary?'). Second, the *indirect inference* of a pattern reveals information, when connecting more than one information source ('The availability pattern of user bunny23 looks suspiciously like the one of my employee John Doe!').

However, Doodle offers users some sort of privacy-enhancing feature. An initiator has the possibility to create "hidden polls" where availability patterns are visible only to him. Together with an SSL based connection, one may achieve confidentiality for external attackers. However, the Doodle server and the poll initiator still learns the availability pattern.

Doodle also reacted to their users demand of security. In a normal poll, everybody may change everybody's vote. If one does not want to register to preserve one's unlinkability, Doodle offers the possibility to restrict changes to a vote from the initiator only. However, trust in the poll initiator and the Doodle server is needed in this case as well.

The outline of the paper is as following: We first raise requirements which we claim to be necessary for privacy-enhanced event scheduling in Section 2. Section 3 will give a short literature overview. Afterwards (Section 4) we discuss which questions are not fulfilled yet and should be target of further research.

2 Requirements

This section describes requirements for privacy-enhanced event scheduling. Even if there may be more general requirements applicable to privacy-enhanced event scheduling, we want to concentrate on the most restrictive ones which still fit the most use cases (but not all). Stating too general requirements might lead to a scheme, which is unnecessarily bloated and inefficient.

An event scheduling application typically schedules an event in a group of a few dozens of people. Even if there are use cases, where scheduling events in a group of people who do not know each other are imaginable,[1] the most common use case of such an application is to schedule an event within a closed group where most of the participants know each other.

A typical web-based event scheduling application can be divided into 3 phases: poll initialization, vote casting and result publication. Figure 1 depicts these 3 phases. An initiator creates a poll in the poll initialization phase. There he has to define a set of possible time slots when an event might take place. Every user may specify his preferences in the vote casting phase. Finally, in the result publication phase, a time slot is chosen, when the event should take place. We call the rule, which chooses the time slot, the *selection rule.*

In a usable privacy-enhanced event scheduling application everything should be the same as in a normal event scheduling application with the difference that the availability pattern is confidential.

The following requirements should apply to a privacy-enhanced event scheduling application:

Untrusted single entity. As little trust as possible should be placed in any single entity.

Verifiability. Every participant should be able to verify that no other participant has cheated (i. e., every other participant is authorized and behaves according the protocol) and that his preferences has been taken into account.

[1] e. g., one needs to schedule a lecture and wants to find out the most acceptable time slot for the students.

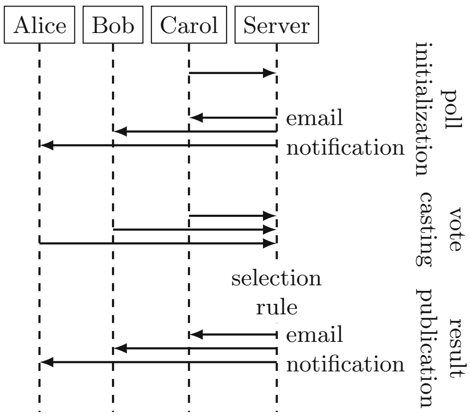

Fig. 1. Phases of a typical web based event scheduling initiated by Carol

Privacy. Nobody should learn more than absolutely necessary about the availability of other participants and thus should not be able to infer on their identity, i. e., every participant should only learn that the one specific chosen time slot fulfills some selection rule. The amount of information which is leaked therefore depends on the selection rule. Note that this requirement does not mean, that all participants are anonymous.

It is easy to conceive, that verifiability and privacy are contradicting requirements. In a Doodle-like scheme without privacy, verifiability is given through the fact, that everybody can read the votes of the other participants. In a privacy-enhanced scheme, other mechanisms are needed.

Additionally, verifiability and privacy require, that the set of participants is fixed within the poll initialization phase and that the participants know each other. Knowing the identity of the other participants gives the possibility to make clear statements about the anonymity set and its size. Additionally one may observe that no attacker is able to add votes for faked participants (only authorized participants vote).

To gain a usable application, the following requirements have to be fulfilled:

Preliminary Steps. An application should not require many preliminary steps (e. g., installation, registration etc.).

Communicational Complexity. An application should not require much more user interaction than existing event schedulers and should allow the user to be offline as it may be used in mobile scenarios. Therefore the scheme has to have few message exchanges. In Figure 1 one can see that typically every user has to interact 2 times with the application: in the vote casting phase and in the result publication. Both interactions are initialized through an email notification.

Computational Complexity. As users do not want to wait long, an application should be efficient for large scheduling problems with many possible event dates.

3 Related Work

Privacy-enhanced event scheduling can be seen as a distributed constraint satisfaction/optimization problem (DSCP/DCOP) or a an e-voting instantiation. We will give a short literature overview of both and explain why they do not fit into the problem. Specific literature is discussed afterwards.

3.1 DSCP/DCOP

A constraint optimization problem consists of a set of variables in finite domains and a set of cost functions. The goal of the optimization is to find a binding for all variables, for which the global cost, calculated from all cost functions is minimal. A constraint satisfaction problem is a special case of the optimization problem where the output of all cost functions is an element of $\{0, 1\}$ and the global cost is calculated with the multiplication of all cost functions. In a distributed constraint optimization, every participant holds his own set of variables and cost functions. The solution to the optimization problem is found through sending messages between the participants with the assignment of variables and their costs.

There exist many algorithms for DCSP [2–4] and DCOP [5–7] and measurements of the information leakage were done by Franzin [8] and Greenstadt [9]. These algorithms may solve complex scheduling problems where different subsets of the participants participate in different events. Each participant may have cost functions about the place, travel time, and constraints between the events.

However, all DCOP algorithms share the problem that they are complex in terms of message exchanges even for basic scenarios. To solve the problem of message exchanges, agents are used which send and receive the messages. As usual users do not want to setup such an agent at some server, they have to run it locally and therefore have to be online at the same time.

Therefore the DCOP approach is too complex in terms of message exchanges and contradicts the communicational complexity requirement. A simpler solution for the simpler problem of scheduling a single meeting would be appropriate.

3.2 E-Voting

There is a lot of literature about electronic voting. It can be categorized into approaches based on mixes [10–14], homomorphic encryption [15–18], and blind signatures [19–23].

The difference between privacy-enhanced event scheduling and e-voting, and why e-voting cannot be applied directly to event scheduling was already discussed [24]. One of the main design criteria of electronic voting schemes is to have a computation and communication complexity, which is independent from the number of participants (voters). While this is valid when organizing an election for millions of citizens, this design criterion can be relaxed in the event scheduling approach, as we deal with smaller closed groups of participants. A design criterion for event scheduling should be that the computational complexity of the scheme scales in the number of time slots. However, if one applies e-voting schemes directly to event scheduling, they all have in common that the number of asymmetric operations scales linear with the number of time slots, i. e., one needs at minimum one asymmetric operation per time slot.

In the following, we will shortly discuss the three approaches separately and estimate the computational complexity in terms of asymmetric operations. Let $|T|$ be the number of possible time slots an event may be scheduled at, and $|P|$ be the number of participants voting in a poll.

Mixes. Chaum invented mixes as building blocks for anonymous communication channels and he first proposed an election scheme based on them [10]. Many extensions have followed thereafter [11–14]. The common idea is to build an anonymous blackboard with mixes. In a first phase, every voter has to generate an asymmetric key pair and publish the public key on that blackboard. In the voting phase, every voter encrypts his or her vote with the secret key and publishes the encrypted vote on the anonymous blackboard. Everybody can decrypt the votes and count the result.

Assuming ℓ mixes, to cast a vote, every voter has to encrypt his or her key and vote ℓ times asymmetrically. A naive adaption to event scheduling would imply one poll per time slot. Every voter would have to do $2 \cdot \ell \cdot |T|$ asymmetric encryptions. Further, one must trust that the mixes do not collude to compromise one's privacy, and the mixes have to perform additional decryption operations, which add to the overall complexity.

Homomorphic Encryption. Voting schemes based on a homomorphic encryption function use the property that one can add all the votes and decrypt the result without decrypting individual votes (i. e., one can find two operations $\oplus$ and $\otimes$ so that the encryption function E is homomorph to these operations $\mathrm{E}(x_1) \oplus \mathrm{E}(x_2) = \mathrm{E}(x_1 \otimes x_2)$). A problem of plain homomorphic encryption is that cheating voters stay undetected as their votes are not decrypted separately. So practical schemes require extra effort to prevent this.

A voting scheme based on a homomorphic encryption function was first described by Cohen and Fischer [15] and later extended by Benaloh and Yung [16]. The scheme consists of different parts where a central entity and the voters have to commit to values and prove their correctness without revealing them. Proving a "primary" ballot is done by committing to several "auxiliary" ones, decrypting half of them and showing that the others are type-equivalent to the primary ballot. This proof is done twice in the whole scheme.

A direct application of this method for event scheduling appears to be inefficient: considering only the vote encryptions, with a security parameter ℓ, every voter would have to do $\ell \cdot |T|$ asymmetric cryptographic operations.

Inspired by Benaloh, Sako and Kilian introduced a voting scheme that uses partially compatible homomorphisms [17]. Baudron et al. enhanced this scheme for multi-votes [18]. However, also in this scheme, a voter has to encrypt every single vote multiple times. Baudron's extension even targets multi-candidate elections, but this is not equivalent to the event scheduling problem, where repeated elections would be needed.

Blind Signatures. Shortly after his mix-approach, Chaum came up with another voting scheme [19] which uses blind signatures [25]. Fujioka et al. reduced the complexity of Chaums idea to adapt it to large scale elections [20] and many subsequent schemes were derived from the Fujioka–Okamoto–Ohta scheme [21–23]. Some of them led to implementations [26, 27].

The main idea is to split the protocol into two independent phases: (1) administration, which handles access control, and (2) counting of anonymously casted votes. The vote is blindly signed during administration, unblinded by the voter and then sent anonymously to the counter. As the casted votes contain no personal information, they can be published afterwards.

If one applies this scheme to the event scheduling problem, a voter would have to blind and unblind $|T|$ messages and verify the administrator's signature of every message. The administrator would have to verify $|P|$ signatures and has to sign $|T|$ messages. The counter would have to verify $|P| \cdot |T|$ signatures. The overall effort is $(|P| + 2) \cdot |T| + |P|$ asymmetric cryptographic operations.

3.3 Specific Prior Work

There are two publications which specifically target the event-scheduling problem [24, 28]. Herlea et al. [28], covers three different approaches to the problem. One is a solution based on a trusted third party, which is efficient, but stands in contrast to the requirement of limited trust in a single entity. The second is a straight application of general secure distributed computing. Consequently, it suffers from high computational and communication complexity. The third approach, a "custom-made negotiation protocol" is most interesting and promising. It can be best described as a hybrid between the techniques reviewed in Section 3. The protocol is designed to schedule single events through a combination of *homomorphic encryption* with respect to the equality operation (in fact, addition modulo two) to *blind* individual availability pattern, and an anonymous channel, which is established by letting voters act as re-encrypting *mixes*. While the cryptographic operations are comparably efficient, the scheme requires way too much communication phases ($|P| + 1$ messages). (The authors acknowledge this and discuss ways to trade off communication complexity against trust assumptions.)

Based on superposed sending and Diffie-Hellman key agreement [29, 30], Kellermann and Böhme described a privacy-enhanced event scheduling solution [24, 31, 32]. Within this solution, every possible participant encrypts his availability pattern with a homomorphic encryption. The availability pattern itself contains a 1 for all available time slots and a 0 for time slots, the participant is unavailable. The homomorphism of the encryption is used afterwards to calculate the number of all available participants at the certain time slots.

4 Open Research Questions

Offering a privacy-enhanced version of a web-based event scheduling system raises several problems which are described in the following and should be target of further research.

4.1 Security Definition and Formal Proof of Correctness

The whole process of multilateral secure voting has assumptions about privacy and security. Security definitions of e-voting or DCOP can already be found in the literature [33]. However, a good application to privacy-enhanced event scheduling as well as a formal proof of correctness is still missing.

A good requirement formalization is still needed. With such a formalization, a proof of correctness of a certain scheme could be done.

4.2 Predefined Complex Selection Rules

A problem a privacy-enhanced event scheduling solution has to deal with is when not all people are available at a certain time slot. The most common selection rule is to choose one of the time slots where most of the people are available. However, it may be desirable to decide on other rules, than solely on the number of available participants. E. g., it may be the case, that some people are more important to be at a meeting than others. Another selection rule may depend on sets of participants, e. g., one may schedule a meeting between a number of organizations where it is necessary that one person of every organization attends the meeting.

When scheduling an event with an existing privacy-invasive solution, one may make such decisions, without the involvement of all possible participants. Even if no common time slot is found where all participants may participate, one is able to select a time slot on other criteria than initially defined. If the selection rule came to no result in a privacy-enhanced scheme one has the problem, that it is not possible to change it without the involvement of all participants. A possible solution may be the specification of more than one selection rule before scheduling an event. The additional selection rules are taken into account when the first one came to no result.

Specifying complex selection rules can be expressed in functions in a DCOP solution. However, e-voting solutions and the specific privacy-enhanced event scheduling approaches suffer these functionality. Here, only the sum of all available participants at the time slots can be taken into account.

A problem is that it is difficult to measure the privacy for an arbitrary selection rule. Measurements which can be found in the DCOP literature may be applied here as well. Furthermore, the amount of privacy has to be displayed to the participants when they state their availabilities. Additionally it is unclear which selection rules with their respecting implications are understandable for average users.

4.3 Specifying Preferences

Within the vote casting phase, every participant has to specify a binary choice if he is available at a certain time slot or not. Instead of sending this binary choice it may be desirable to specify preferences for every time slot. This can already be done in a Doodle poll. There one may create a so called "Yes-No-Ifneedbe"-poll where one has the possibility to select one out of three states for every time slot.

Such a generalization is not necessarily possible in a privacy-enhanced event scheduling solution. E. g., when using homomorphic encryption the votes cannot be taken from an arbitrary domain. Consequently, one may not apply arbitrary operations to the votes. This should be target of further research and one may think of either giving the answers a meaning (like Doodle does) or let the user specify a preference value.

4.4 Prevent "Legal-but-Selfish Votes"

A legal-but-selfish vote is a vote where a participant indicates his availability only at his preferred time slot and not at all time slots he is available. Solutions which only come to a solution if all participants are available at a common time slot (e. g., DCSP) motivate users to anonymously send legal-but-selfish votes.

Within a scheduling solution where all participants can observe all indicated availabilities, the motivation of being selfish would go along with reputation loss. However, in a privacy-friendly solution selfish votes may be sent anonymously.

A possible prevention of selfish votes may let participants prove that they signaled availability for more than a certain minimum number of time slots. Knowing the fact that every voter must be available at a number time slots in order to participate at the poll would of course decrease the privacy of the voters.

4.5 Automatic Poll Termination

It was already mentioned, that it is necessary to fix the set of participants in the poll initialization phase. However, it might occur that a participant should be removed after a poll has started.

Kellermann and Böhme already discussed how to kick-out users who should not participate in the poll anymore. From a usability point of view, this seems to be a valid requirement, as depending on the scheme, the selection rule might be unable to calculate the result, if one of the participants does not vote. If one thinks one step ahead, an open question is how to handle an automatic termination of the poll, either after a specific amount of time, or after a specific number of participants voted. Such an extension should not undermine the privacy of the participants.

4.6 Dynamic Insertion and Deletion of Time Slots

Sometimes it may be necessary to dynamically insert or delete time slots, after the poll has already started. This may occur after participants already casted votes.

Dynamic insertion and deletion of time slots may not be obviously possible in a privacy-enhanced event-scheduling scheme (e. g., it may influence the cost functions of other time slots in a DCOP based solution). If one does, already casted votes should be taken into account. The privacy of votes for all time slots should not be affected, when the set of time slots is changed. Current schemes do not target this issue.

4.7 Updating and Revoking Votes

A feature offered by event scheduling applications is that one has the possibility to change or even delete one's vote at any point of time. This is not necessarily possible in an easy way within a privacy-enhanced solution.

Changing a vote within the vote casting phase might help an attacker to undermine the privacy of participants (e. g., it may release release information about the secret key in homomorphic schemes). Completely deleting votes might make the selection process impossible as it may be the case that all votes are needed for the selection rule.

5 Conclusion

We formulated requirements of privacy-enhanced event scheduling and gave a short literature overview of existing solutions. Open research questions belonging to privacy-enhanced event scheduling where raised. Some of the research questions where already fulfilled in existing schemes (e. g., DCOP may solve complex selection rules and specify preferences; there is no motivation of legal-but-selfish votes when using e-voting schemes). However, to the best of our knowledge there is neither a scheme which is designed to be usable nor an application which offers usable privacy-enhanced event scheduling currently.

Acknowledgments. The author thanks Stefanie Pötzsch and Sandra Steinbrecher for their valuable suggestions.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreement № 216483.

References

1. Näf, M.: Doodle homepage (April 2010), `http://www.doodle.com`
2. Silaghi, M.C., Sam-Haroud, D., Faltings, B.: Asynchronous search with aggregations. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pp. 917–922. AAAI Press/The MIT Press (2000)
3. Yokoo, M., Hirayama, K.: Algorithms for distributed constraint satisfaction: A review. Autonomous Agents and Multi-Agent Systems 3(2), 185–207 (2000)
4. Léauté, T., Faltings, B.: Privacy-preserving multi-agent constraint satisfaction. In: Conference on Information Privacy, Security, Risk and Trust (PASSAT 2009) [34], pp. 17–25 (2009)
5. Modi, P.J., Shen, W.M., Tambe, M., Yokoo, M.: Adopt: Asynchronous distributed constraint optimization with quality guarantees. Artificial Intelligence 161, 149–180 (2004)

6. Maheswaran, R.T., Tambe, M., Bowring, E., Pearce, J.P., Varakantham, P.: Taking dcop to the real world: Efficient complete solutions for distributed multi-event scheduling. In: AAMAS, pp. 310–317. IEEE Computer Society, Los Alamitos (2004)
7. Mailler, R., Lesser, V.: Solving distributed constraint optimization problems using cooperative mediation. In: AAMAS 2004: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, Washington, DC, USA, pp. 438–445. IEEE Computer Society, Los Alamitos (2004)
8. Franzin, M.S., Freuder, E.C., Rossi, F., Wallace, R.: Multi-agent meeting scheduling with preferences: Efficiency, privacy loss, and solution quality. AAAI Technical Report WS-02-13 (2002)
9. Greenstadt, R., Pearce, J.P., Bowring, E., Tambe, M.: Experimental analysis of privacy loss in DCOP algorithms. In: Proc. of ACM AAMAS, pp. 1424–1426. ACM Press, New York (2006)
10. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. ACM Commun. 24(2), 84–90 (1981)
11. Park, C., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/Nothing election scheme. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994)
12. Ogata, W., Kurosawa, K., Sako, K., Takatani, K.: Fault tolerant anonymous channel. In: Han, Y., Okamoto, T., Qing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 440–444. Springer, Heidelberg (1997)
13. Abe, M.: Universally verifiable mix-net with verification work independent of the number of mix-servers. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 437–447. Springer, Heidelberg (1998)
14. Jakobsson, M.: A practical mix. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 448–461. Springer, Heidelberg (1998)
15. Cohen, J.D., Fischer, M.J.: A robust and verifiable cryptographically secure election scheme. In: SFCS 1985: Proceedings of the 26th Annual Symposium on Foundations of Computer Science (sfcs 1985), Washington, DC, USA, pp. 372–382. IEEE Computer Society, Los Alamitos (1985)
16. Benaloh, J.C., Yung, M.: Distributing the power of a government to enhance the privacy of voters. In: PODC 1986: Proceedings of the Fifth Annual ACM Symposium on Principles of Distributed Computing, pp. 52–62. ACM, New York (1986)
17. Sako, K., Kilian, J.: Secure voting using partially compatible homomorphisms. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 411–424. Springer, Heidelberg (1994)
18. Baudron, O., Fouque, P.A., Pointcheval, D., Stern, J., Poupard, G.: Practical multi-candidate election system. In: PODC 2001: Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing, pp. 274–283. ACM, New York (2001)
19. Chaum, D.: Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 177–182. Springer, Heidelberg (1988)
20. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Seberry, J., Zheng, Y. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1992)
21. Sako, K.: Electronic voting scheme allowing open objection to the tally. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 77(1), 24–30 (1994)

22. Ohkubo, M., Miura, F., Abe, M., Fujioka, A., Okamoto, T.: An improvement on a practical secret voting scheme. In: Mambo, M., Zheng, Y. (eds.) ISW 1999. LNCS, vol. 1729, pp. 225–234. Springer, Heidelberg (1999)
23. DuRette, B.W.: Multiple administrators for electronic voting. Bachelor's thesis, Massachusetts Institute of Technology (May 1999)
24. Kellermann, B., Böhme, R.: Privacy-enhanced event scheduling. In: Conference on Information Privacy, Security, Risk and Trust, PASSAT 2009 [34], pp. 52–59 (2009)
25. Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. ACM Commun. 28(10), 1030–1044 (1985)
26. Cranor, L., Cytron, R.: Sensus: A security-conscious electronic polling system for the internet (1997)
27. Herschberg, M.A.: Secure electronic voting over the world wide web. Master's thesis, Massachusetts Institute of Technology (May 1997)
28. Herlea, T., Claessens, J., Preneel, B., Neven, G., Piessens, F., Decker, B.D.: On securely scheduling a meeting. In: Dupuy, M., Paradinas, P. (eds.) SEC. IFIP Conference Proceedings, vol. 193, pp. 183–198. Kluwer, Dordrecht (2001)
29. Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of Cryptology 1(1), 65–75 (1988)
30. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory IT-22(6), 644–654 (1976)
31. Kellermann, B.: Datenschutzfreundliche Terminplanung. In: Mehldau, M.w. (ed.) Proceedings of the 26th Chaos Communication Congress, Marktstraße 18, 33602 Bielefeld, Chaos Computer Club, Art d'Ameublement, pp. 207–211 (December 2009)
32. Kellermann, B.: Dudle homepage (April 2010), `http://dudle.inf.tu-dresden.de`
33. Greenstadt, R., Smith, M.D.: Collaborative scheduling: Threats and promises. In: Workshop on Economics and Information Security (2006)
34. IEEE/IFIP: Proceedings of IEEE International Conference on Computational Science and Engineering, Conference on Information Privacy, Security, Risk and Trust (PASSAT 2009), Los Alamitos, CA, USA, IEEE/IFIP, vol. 3. IEEE Computer Society (2009)