Privacy-Enhanced Event Scheduling with Majority Agreement

Benjamin Kellermann
Technische Universität Dresden
Faculty of Computer Science
D-01062 Dresden, Germany
Benjamin.Kellermann@tu-dresden.de

ABSTRACT

Applications which help users to schedule events are becoming more and more important. They all have in common, that the preferences of all participants are revealed to the other participants. Kellermann and Böhme described a privacy-enhanced solution of an event scheduling protocol. This protocol has one major drawback. A meeting is only scheduled, if all participants are available at a common time slot (unanimous agreement).

Within our paper, we propose an extension to this protocol, which overcomes the restriction of unanimous agreement. With this extension, it is possible to use less strict rules, i. e., the decision can be made from the number of available participants. The most commonly used decision therefore would be to choose the time slot, which the maximum number of participants agrees on. An implementation of the protocol, as well as some performance measurements are presented as well, which shows that the protocol is practical feasible.

Keywords

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

1. INTRODUCTION

Since Web 2.0 applications became more and more popular, some special purpose applications like Doodle [30] have received much attention as well. This application lets users create polls to "schedule an event such as a board meeting, business lunch, conference call, family reunion, movie night, or any other group event" [30]. To schedule an event, three steps are necessary: create a poll, cast a vote, and close the poll by choosing the final date. Within vote casting, every participant submits his so called *availability pattern* to the Doodle server. As privacy and security is only a secondary goal and to keep the application simple and easy, everybody is allowed to view and edit votes already cast. When closing the poll, a row summing up the individual availabilities is displayed to help with the date selection.

Even the Doodle developers have recognized that users demand some privacy and security. One may create "hidden polls" where availability patterns are visible to the initiator only. To achieve confidentiality for the connection, one may use encrypted access via SSL. One may create accounts and have username/password-based access control to the votes. However, complete trust in the Doodle server and the poll initiator is needed in all cases.

Additionally to event scheduling, it is possible to "make a choice among movies, menus, travel destinations, or among any other selection" [30] with such applications. Ignoring privacy constraints in such polls might release sensitive personal information and may influence the freedom of choice of the participants.

Kellermann and Böhme proposed a privacy-friendly and verifiable solution for scheduling a single event [24]. It hides availability patterns and reveals only the sum of available participants at every time slot. Additionally, the computational complexity is independent of the number of available time slots. However, it has the drawback that unanimous agreement is required. As long as meetings in very small groups are scheduled, this drawback might be acceptable. However, when scheduling meetings with larger sets of participants (e. g., 10 to 20), it will become difficult to find a common time slot. Additionally, unanimous agreement combined with anonymous vote casting encourages participants to cast so called "legal-but-selfish" votes, where a participant indicates unavailability at all time slots except the one he wants to win.

In this paper we present an extension to the scheme, which overcomes the strict selection rule of unanimous agreement. After introducing the notation and stating requirements for privacy-enhanced event scheduling in Section 2, we discuss related work in Section 3. Section 4 describe the two main attacks to the old scheme and to which restrictions they lead there. How these two attacks can be prevented without the need of unanimity is shown in Section 5 and 6. An implementation of the protocol is presented in Section 7. Readers who want to have a look at the whole protocol should refer to Appendix A.

2. NOTATION AND REQUIREMENTS

2.1 Notation

We will use the following terminology and notation:

Initiator The organizer of an event, a person who sets up a poll. This person may also participate in the poll as a participant.

Participant (u) A participant (user) in the poll. U is the ordered set of all participants so that $|U|$ is the number of participants in the poll.

Time slot (t) A specific date and time at which an event can take place. The set of time slots is denoted by T.

Available participants (σ_t) The sum of available participants at time slot t.

Selection rule A function which decides about which time slot is chosen.

Anonymity set A set of participants, within which a specific participant is not identifiable [34].

2.2 Requirements

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 all participants know each other. The following requirements should apply to a privacy-enhanced event scheduling application [17, 23]:

Verifiability Every participant should be able to verify that no other participant has cheated and that his vote has been counted.

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 the selection rule.

Untrusted single entity As we do not know anything about a possible attacker in advance, the system should consider minimal assumptions about the attacker. Because of that, as little trust as possible should be placed in any single entity.

Usability An application should not require much more user interaction than existing event schedulers (e. g., message exchanges, program installation etc.).

Efficiency The scheme should be efficient for large scheduling problems with many possible event dates.

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

3. RELATED WORK

There are several approaches dealing with event scheduling. It can be seen as distributed constraint satisfaction / optimization problem (DCSP/DCOP) or as an instance of electronic voting. We will discuss DCSP/DCOP approaches first (Section 3.1) and look at e-voting afterwards (Section 3.2). In Section 3.3 we will discuss specific publications about single event scheduling.

3.1 DCSP/DCOP

A constraint optimization problem consists of a set of variables within finite domains and a set of cost functions. The goal of the optimization is to find an assignment of the variables so that the global costs are minimized. Solving this problem in a distributed way means that each participant holds its own set of variables and cost functions. The solution to the problem is found by exchanging messages with assignments to variables and their costs between the participants.

There exist many algorithms for DCSP [25, 39, 41] and DCOP [26–28] and measurements of the information leakage were done by Franzin [14] and Greenstadt [16]. With the help of these algorithms, complex scheduling problems may be solved (e. g., scheduling of many events, where different subsets of the participants participate in each event with constraints about place, travel time etc.). 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 have to be online at the same time.

Therefore, the DCOP approach is too complex in terms of message exchanges and 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 [1, 9, 21, 31, 33], homomorphic encryption [5, 6, 11, 37], and blind signatures [8, 13, 15, 32, 36].

The difference between privacy-enhanced event scheduling and e-voting, and why e-voting cannot be applied directly to event scheduling has been discussed already [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). This is a valid assumption, as an e-voting scheme should run in a scenario with millions of voters. This design criterion can be relaxed in our 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. For example, it may be desirable to agree on a 1 hour meeting within the next two weeks. As a typical working-week has 40 hours, assuming that meetings are aligned to full hours, we would end up with 40 votes per week.

Table 1: Execution time for an exponentiation modulo a 786 bit long integer in JavaScript on an Intel Pentium 4 Duo with 2.8 GHz, 2 GB RAM running Windows XP SP3 with different libraries and browsers

	Wu [40]	Leemon [4]	Shapiro [38]
Internet Explorer 8	2.80 s	5.09 s	(crashes)
Firefox 3.6.6	0.79 s	0.82 s	6.65 s
Safari 5.0	0.90 s	0.15 s	4.59 s
Opera 10.60	0.31 s	0.18 s	3.39 s
Google Chrome 5.0	0.17 s	0.25 s	1.36 s

Assuming possible start dates at every quarter of the hour, we would end up with $40 \cdot 4 = 160$ starting times per week. This totals to 320 different starting times which all require a binary vote. Current implementations like the general doodle interface certainly run into usability problems when offering polls with 320 options. But it is easy to conceive interfaces to personal electronic calendars, similar to common office groupware, and in fact doodle implemented such an interface recently [29].

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. Let us assume, we want to build a Web 2.0 application which should run completely within a browser. To perform some cryptography on client side, a programmer has two options to use as a programming language: JavaScript and Flash. There exist BigInteger libraries for both programming languages. However, JavaScript is available at more browsers and platforms. We implemented a performance measurement for a discrete exponentiation modulo a 786 bit long integer. We measured the execution time which is needed with different public available libraries on the most common used browsers. We tried all public libraries we found, and measured the ones from Wu [40], Leemon [4], and Shapiro [38].[2] Table 1 shows how much time is needed for one asymmetric operation. One can easily see that wasting asymmetric operations would result in a large time penalty and would make an application unusable. I. e., assuming only one asymmetric operation per time slot using the library of Wu would need approx. $2.8\,\text{s} \cdot 320 \approx 15\,\text{min}$ of computation in our example above. Having a low computation complexity is even more important considering small mobile devices like smartphones which are much more limited in speed and energy.

Web-based e-voting systems found in the literature are E-Vox [18] and Helios [2]. However, E-Vox seems to be no longer available and Helios crashed when we tried to configure even a simple poll [3]. In addition, poll verification is done with a Python script in case of Helios Voting, which is mentioned to need about 4 hours for poll auditing [2]. Another implementation is the Java-based Civitas [10]. This implementation currently does not work in a web-based manner, but might be turned into an applet in the future.

[2]The table shows only 3 of the 5 investigated libraries. The remaining two were too slow to be mentioned here. Note, that we didn't investigate, why the library from Shapiro crashes on Internet Explorer, as it was the slowest of the 3 remaining anyway.

3.3 Single Event Scheduling

Herlea et al. proposed a "custom-made negotiation protocol" to secure and verifiable event scheduling [17], which they implemented in a prototype named agenTa. It can be described as a hybrid technique between of homomorphic encryption, blind signatures and mixes. The protocol is designed to schedule single events through a combination of homomorphic encryption with respect to the equality operation (addition modulo two) to blind individual availability patterns, and an anonymous channel, which is established by letting voters act as re-encrypting mixes. While having efficient cryptographic operations, they need many communication phases. The authors acknowledge this and discuss ways to trade off communication complexity against trust assumptions. In addition, only unanimous agreement is achieved, which is the property we want to overcome.

Another protocol was proposed by Kellermann and Böhme [24]. The main idea of the protocol is to use superposed sending, generalized to other Abelian groups than GF(2) [7]. A dedicated DC-Net round is executed for every nominated time slot. Each participant sends an encrypted 1 in the specific round if he is available at the time slot, and 0 otherwise. Through the built-in homomorphism, the sum of the votes is calculated. Therefore, the result of one DC-Net round is the number of available participants at this time slot.

Because of using a DC-Net per time slot, every participant needs a key with every other participant for every DC-Net. To be efficient, the authors propose to use Diffie-Hellman key agreement [12] to calculate one seed with every other participant. This seed is used for all DC-Nets and furthermore for all future polls. Except one additional signature, all other operations in the scheme are symmetric.

4. LIMITATIONS OF THE OLD SCHEME

In the following, we will call the original scheme from Kellermann and Böhme the "old scheme". The main problem of using superposed sending, generalized to other Abelian groups than GF(2) is that a participant may send values different from 0 or 1. I. e., a participant may send values below 0 to lower the chance for a specific time slot of being chosen, and values above 1 to increase it. An example of this is illustrated in Figure 1. The tables show polls with 3 participants and 4 time slots. Let Alice be u_a, Bob u_b, and Mallory u_m. The votes $(v_{u,t})$ are shown inside each table. Mallory tries to manipulate the poll in a way that time slot t_3 would win. In Figure 1a she tries to decrease the sum of t_1 with sending a -1. Figure 1b shows another attack where she sends a $+2$ at t_3. In both polls the column of t_3 results in the largest sum. Because of the anonymization of all messages through the DC-Net, Mallory's attack is hidden to Alice and Bob.

Kellermann and Böhme introduced two mechanisms to overcome this problem:

	t_0	t_1	t_2	t_3
Alice	0	1	0	0
Bob	1	1	0	1
Mallory	0	−1	0	1
$\sum$	1	1	0	2

(a) attacking with −1

	t_0	t_1	t_2	t_3
Alice	0	1	0	0
Bob	1	1	0	1
Mallory	0	0	0	2
$\sum$	1	2	0	3

(b) attacking with +2

Figure 1: Different ways to attack a poll when Mallory wants t_3 to win. The plain text values of the votes ($v_{u,t}$) are displayed.

1. Unanimous agreement makes attacks with values lower than 0 unnecessary and attacks with values higher than 1 visible. As only time slots are accepted where all participants are available, sending a 0 at some time slot is enough to rule out the chance for the time slot of being chosen.
   Every participant checks if he voted for the chosen time slot. If any time slot is chosen where a participant has not vote for, somebody else sent a value higher than 1. Therefore, if everybody checks if he voted for the chosen time slot attacks with values higher than 1 will be detected.[3]
2. An optional verification phase was introduced where possible cheaters are unmasked and attacks to the availability of the system are made unattractive.

While the scheme works well with these two methods, it is very inflexible due to the strongly restricted selection rule. In addition it encourages attackers to send "legal-but-selfish" votes, where Mallory would send a 0 at all time slots but the one she wants to win (cp. Figure 2a).

In the following, we will discuss how to extend the scheme in a way that sending values different from 0 or 1 can be prevented without the need of unanimous agreement. Dropping this restriction, an arbitrary selection rule which operates on the sums of available participants can be used (e. g., the time slot where most participants are available). We first discuss how attacks with values lower than 0 (i. e., (−1)-attacks) are prevented (Section 5) and look at attacks with higher values (i. e., (+2)-attacks) later (Section 6).

5. PREVENTING (−1)-ATTACKS

We already showed a (−1)-attack in Figure 1a. Another example is shown in Figure 2b where Mallory tries to attack the poll more naively. There, Mallory sends a −1 at all time slots she does not want to win ($v_{u_m,t_0} = v_{u_m,t_1} = v_{u_m,t_2} = -1$). This attack can be detected easily at time slot t_2, where the result is −1. The result of every time slot should be a value between 0 and the number of all participants, which is 3 in this example. In our extension, we use the fact that the result of a vote at some time slot cannot be lower than 0.

Instead of using one dedicated DC-Net round for every time slot, we propose to use several simultaneously running DC-Net rounds. Let I be the number of simultaneously running DC-Net rounds. Every participant u splits his vote $v_{u,t} \in \{0,1\}$ which contains his availability at time slot t into I partial votes $\bar{v}_{u,t,0}, \ldots, \bar{v}_{u,t,I-1}$ such that:

[3]In Figure 1b Alice would recognize that there occurred an attack as she send a 0 at the time slot which was chosen.

	t_0	t_1	t_2	t_3
Alice	0	1	0	0
Bob	1	1	0	1
Mallory	0	0	0	1
$\sum$	1	2	0	2

(a) legal-but-selfish vote

	t_0	t_1	t_2	t_3
Alice	0	1	0	0
Bob	1	1	0	1
Mallory	−1	−1	−1	1
$\sum$	0	1	−1	2

(b) naive attack

Figure 2: Two examples what could happen when Mallory wants t_3 to win. Sending a legal-but-selfish vote (a) would not help t_3 to win with certainty. Sending a −1 at all other time slots than t_3 (b) would be detected at t_2 at least, as the sum should be an element of $\{0, \ldots, |U|\}$.

1. An index $j \in \mathbb{Z}_I$ for one partial vote is chosen randomly and kept secret.
2. The partial vote with index j ($\bar{v}_{u,t,j}$) is equal to the participants' actual vote $v_{u,t}$.
3. The remaining $I - 1$ partial votes are equal to 0.

If all participants are honest, the following properties result from the construction:

1. For all time slots $t \in T$ and partial vote indices $i \in \mathbb{Z}_I$, the sum of all partial votes of all participants is an element between 0 and the number of participants ($\forall t, i : \sum_{u \in U} \bar{v}_{u,t,i} \in \{0, \ldots, |U|\}$).
2. At one time slot, the sum of all partial votes of all participants is the sum of all available participants at this time slot ($\sigma_t = \sum_{u \in U} \sum_{i=0}^{I-1} \bar{v}_{u,t,i}$).

In Figure 2b, we have seen that an attacker has to guess where a honest participant will send a 1. With this extension it is not enough for Mallory to guess the availability of another participant, she further has to guess at which partial vote the actual vote was sent. This is difficult as long as the chosen partial vote index is random, kept secret, and the number of partial votes per time slot $\mathbb{Z}_I$ is sufficiently high.

Figure 3 shows an example of the vote vector splitting with $I = 3$, where Mallory tries to cheat in the same way as already shown in Figure 1a. Mallory sends a −1 at time slot t_1 ($v_{u_m,t_1} = -1$). The left table shows the old scheme. There the attack would remain undetected as all elements of the result vector are in the allowed range. The right table shows the same vote vectors split into several vectors. There, for time slot t_1 Alice has chosen the first table ($i = 0$) for her vote ($\bar{v}_{u_a,t_1,0} = v_{u_a,t_1} = 1$) and Bob has chosen the second one. As Mallory has chosen the third table ($i = 2$) her attack can be detected.

	t_0	t_1	t_2	t_3
Alice	0	1	0	0
Bob	1	1	0	1
Mallory	0	−1	0	1
$\sum$	1	1	0	2

	t_0	t_1	t_2	t_3	
Alice	0	1	0	0	$i=0$
Bob	0	0	0	0	
Mallory	0	0	0	0	
$\sum$	0	1	0	0	
Alice	0	0	0	0	$i=1$
Bob	0	1	0	1	
Mallory	0	0	0	1	
$\sum$	0	1	0	2	
Alice	0	0	0	0	$i=2$
Bob	1	0	0	0	
Mallory	0	−1	0	0	
$\sum$	1	−1	0	0	
$\sum$	1	1	0	2	

Figure 3: Split the votes into several partial votes. While Mallory's attack remains undetected in the left table, Alice and Bob are able to detect it in the right one.

In the following, we want to compare this extension of preventing (-1)-attacks with our requirements verifiability and privacy stated in Section 2.2. A discussion of the other requirements is done in Section 6.

5.1 Verifiability

When verifying that no attack occurred, we can distinguish two cases:

- In the simpler one, one DC-Net round for one time slot and one partial vote index results -1 ($\sum_{u\in U} \bar{d}_{u,t,i} = -1$). This case occurred in the example in Figure 3.
- In a more complex scenario, some participant sent a 1 in some DC-Net round, but the result equals 0 due to a (-1)-attack. This may also occur in the old scheme. Bob for example can detect that Mallory has cheated at t_0 in Figure 2b. However, to prove that Mallory has cheated at t_0, Bob has to give up his privacy. In such a case, Bob can decide for himself what is worth more, his privacy or to unmask Mallory.

In the following, we first discuss the case where the privacy is the most valuable good, and we discuss situations later where participants are willing to give up their privacy for unmasking attackers. For reasons of simplicity, we write all calculations which are done in the DC-Net without the modulo operations. Additionally, we consider only one attacker.

5.1.1 *Without Privacy-Loss*

Checking the correctness of a poll can be expressed by a function. It takes all messages sent within the poll and returns true if the poll was correct, and false otherwise $\mathcal{V} : \mathbb{Z}^{|U|\times|T|\times I} \to \{\text{true}, \text{false}\}$. Let $\bar{\boldsymbol{d}} \in \mathbb{Z}^{|U|\times|T|\times I}$ be the 3-dimensional array of all DC-Net messages containing elements of $\bar{d}_{u,t,i}$ for a DC-Net message from participant u at time slot t and partial vote index i. The function which checks the correctness of the poll is defined as:

$$\mathcal{V}(\bar{\boldsymbol{d}}) = \begin{cases} \text{true} & \text{if } \forall t \in T, i \in \mathbb{Z}_I : \sum_{u\in U} \bar{d}_{u,t,i} \in \{0,\dots,|U|\} \\ \text{false} & \text{otherwise.} \end{cases} \tag{1}$$

We assume one attacker Mallory (u_m) who tries to send a -1 at time slot t. Assuming x participants voting for t, the probability of detecting Mallory's attack is calculated by

$$P(\mathcal{V}(\bar{\boldsymbol{d}}) = \text{false} \mid v_{u_m,t} = -1) = \left(\frac{I-1}{I}\right)^x. \tag{2}$$

With an increasing x, the probability of detection would decrease. In a worst case scenario w. r. t. detecting attackers all honest participants send a 1 for all time slots ($x = |U| - 1$) and therefore the lower bound of the probability to detect the attack is

$$P(\mathcal{V}(\bar{\boldsymbol{d}}) = \text{false} \mid v_{u_m,t} = -1) \geq \left(\frac{I-1}{I}\right)^{|U|-1}. \tag{3}$$

The probability of successfully performing an attack would increase with an increasing number of participants. Therefore, one should choose the number of DC-Net rounds I dependent on the number of participants $|U|$.

If Mallory tries to send a -2 the chance of detection increases. In such a case she needs to find two DC-Nets where her attack can be covered.[4] Calculating this probability can be expressed in the urn model with $|U|$ balls. Mallory colors two balls with different colors (the two DC-Nets where she sends her -1). One ball is drawn and returned for every honest participant. If both colored balls occur at least once in the drawing, Mallory's attack will remain undetected. The attack would be detected in three cases:

c_1: None of Mallory's balls occur in the output.

c_2: The first ball occurs at least once in the output, but not the second.

c_3: The second but not the first ball occurs.

If all $|U| - 1$ honest participants vote for the attacked time slot, the probability of the first case will reach a minimum. It can be calculated by

$$P(c_1) \geq \left(\frac{I-2}{I}\right)^{|U|-1}. \tag{4}$$

The probability, that one ball does not occur is $\left(\frac{I-1}{I}\right)^{|U|-1}$ and to calculate the probabilities for the second and third case, we have to subtract $P(c_1)$ from this to ensure that the other ball occurs:

$$P(c_2) = P(c_3) \geq \left(\frac{I-1}{I}\right)^{|U|-1} - P(c_1) \tag{5}$$

[4]Note that Mallory may also choose one DC-Net for sending a -2. However, the chance for staying undetected is better if she chooses different DC-Nets and therefore we only stick to this case.

Summing up the probability of detecting Mallory's attack is calculated by adding all three probabilities:

$$P(\mathcal{V}(\bar{\boldsymbol{d}}) = \text{false} \mid v_{u,t} = -2) \geq 2 \cdot \left(\frac{I-1}{I}\right)^{|U|-1} - \left(\frac{I-2}{I}\right)^{|U|-1}. \quad (6)$$

Generalized to sending the value $-n$ this probability can be considered as a multinomial distribution but a general formula is out of the scope of this paper.

5.1.2 With Privacy-Loss

We already discussed that a person who sent a 1 in a DC-Net which results in 0 is in the position to unmask the attacker with the drawback of giving up his privacy.[5] For such a case, we can define another function checking the correctness of the poll. This function will return false if there exists a participant u_p, who can prove that he sent a 1 at a time slot t and DC-Net round with partial vote index i which resulted in 0 ($\sum_{u \in U} \bar{d}_{u,t,i} = 0$).[6] Let $\bar{\boldsymbol{k}} \in \mathbb{Z}^{|U| \times (|U|-1) \times |T| \times I}$ be the 4-dimensional array of all keys used in all DC-Nets of the poll. The function which checks the correctness of the poll under the assumption that all participants are willing to disclose their availability at one time slot to unmask an attacker is defined as

$$\mathcal{B}\left(\bar{\boldsymbol{d}}, \bar{\boldsymbol{k}}\right) = \begin{cases} \text{true} & \text{if } \neg\exists u_p \in U, t \in T, i \in \mathbb{Z}_I : \\ & \quad \left(\sum_{u \in U} \bar{d}_{u,t,i} = 0\right) \wedge \\ & \quad \left(\bar{d}_{u_p,t,i} + \sum_{u \in U, u \neq u_p} \bar{k}_{u_p,u,t,i} = 1\right) \\ \text{false} & \text{otherwise.} \end{cases} \quad (7)$$

To run an undetected -1-attack under these assumptions, Mallory needs two honest participants sending a 1 in the same DC-Net round to hide her attack.[7] The probability that this attack is detected is calculated by the addition of the probabilities of the two cases

c_1: nobody choses Mallory's DC-Net, and

c_2: one participant choses Mallory's DC-Net.

For c_1, the sum of the attacked DC-Net will be -1 and therefore $\mathcal{V}(\bar{\boldsymbol{d}})$ will fail (see Equation 3). The lower bound[8] for the probability of the second case is the probability where the output of $\mathcal{B}\left(\bar{\boldsymbol{d}}, \bar{\boldsymbol{k}}\right)$ is false and can be calculated with

$$P(\mathcal{B}\left(\bar{\boldsymbol{d}}, \bar{\boldsymbol{k}}\right) = \text{false} \mid v_{u,t} = -1) \geq (|U|-1) \cdot \frac{1}{I} \cdot \left(\frac{I-1}{I}\right)^{|U|-2}. \quad (8)$$

[5]E. g., Bob could detect that Mallory cheated at t_0 in Figure 2b.

[6] If this sum is lower than 0, an attack occurred as well. However, as this attack would be discovered by function $\mathcal{V}(\bar{\boldsymbol{d}})$ (Equation 1), we want to neglect this case here.

[7]This is like trying to perform a -2-attack without privacy-loss, but choosing one partial DC-Net to send the -2.

[8]All $|U|-1$ honest participants voted for the attacked time slot

Table 2: Lower bounds for the probability of successfully detecting an attack

I	$\lvert U\rvert$	without giving up privacy $P(\mathcal{V}(\bar{\boldsymbol{d}}) = \text{false} \mid v_{u,t} = -1)$	without giving up privacy $P(\mathcal{V}(\bar{\boldsymbol{d}}) = \text{false} \mid v_{u,t} = -2)$	with privacyloss $P(\mathcal{V}(\bar{\boldsymbol{d}}) = \text{false} \vee \mathcal{B}(\bar{\boldsymbol{d}}, \bar{\boldsymbol{k}}) = \text{false} \mid v_{u,t} = -1)$	probability of privacyloss $P(\mathcal{B}(\bar{\boldsymbol{d}}, \bar{\boldsymbol{k}}) = \text{false} \mid v_{u,t} = -1)$
20	15	48.8 %	74.7 %	84.7 %	(35.9 %)
20	5	81.5 %	97.3 %	98.6 %	(17.1 %)
50	15	75.4 %	94.3 %	96.9 %	(21.5 %)
50	5	92.2 %	99.5 %	99.8 %	(7.5 %)
100	15	86.9 %	98.4 %	99.2 %	(12.3 %)
100	5	96.1 %	99.9 %	99.9 %	(3.9 %)

Note that it is not possible, that $\mathcal{V}(\bar{\boldsymbol{d}}) = \text{true}$ and $\mathcal{B}\left(\bar{\boldsymbol{d}}, \bar{\boldsymbol{k}}\right) = \text{true}$ (cp. Footnote 6) and therefore we can add the Probabilities of Equations 6 and 8 to get the overall probability of detecting a (-1)-attack if users are willing to disclose their availability to unmask attackers.

5.1.3 Summary

Table 2 illustrates these formulas with some example values. One can see that splitting the vote vector into 20 partial vote vectors makes it rather unlikely to perform an undetected attack against small polls with 5 participants. The chance to detect a (-1)-vote at this time slot is at least[9] 81.5 %. Additionally to these 81.5 %, the attack can be discovered with a probability 17.1 % by one of the participants. If all participants are willing to disclose their availability at the attacked time slot, the detection probability is at least 98.6 %.

In case of $\mathcal{V}(\bar{\boldsymbol{d}}) = \text{false}$, the decryption of the DC-Net round can be requested where the invalid value occured. Therefore every participant has to reveal his key for the DC-Net round. The single votes can be decrypted with the keys which identifies the attacker. The attacker may modify his key to hide his attack in this phase. However, this can be prevented in the same way as it was proposed for the verification phase of the old scheme (see Appendix A.5 for details).

However, if availabilities should not be disclosed under any circumstances, the attacker identification may be skipped. One has to accept in this case that attackers are able to attack the availability of a poll anonymously. Then one may decide with function $\mathcal{V}(\bar{\boldsymbol{d}})$ (Equation 1) that some attack occurred, but skip revealing keys to avoid possible decryption of votes. Note that the decision to perform a poll with privacy loss

[9]if all 4 honest participants vote for a time slot

or without has to be accepted by all participants. If every participant may decide on his own, an attacker will always refuse to reveal his keys for one DC-Net round, stating she has to cover some vote.

If a participant discovers an attack with the function $\mathcal{B}\left(\bar{\boldsymbol{d}}, \bar{\boldsymbol{k}}\right)$, he may decide on his own if he gives up his privacy to unmask the attacker. Therefore, the lower bound for the probability of detecting an attack is some value between both lower bounds, depending if the actual participant which detected the attack is willing to reveal his vote.

5.2 Privacy

The possible decryption in the verification phase to detect the attacker may be a privacy problem. The old scheme had the same decryption in the verification phase. However, unlike the old scheme, the probability that this really is a privacy problem is very low, as the attacker has to guess the index for the DC-Net round where the victim sent his vote. The probability of guessing the index of a specific victim is $\frac{1}{I}$.

Attacking more than one DC-Net round with negative values to increase the probability of hitting the victim's DC-Net does not help the attacker, because the goal of the honest participants is to find only one DC-Net which was attacked. Therefore, it is enough to disclose the keys for one attacked round. The algorithm to choose the DC-Net round to be disclosed should choose one of the rounds with the lowest sum. The function $\mathcal{D} : \mathbb{Z}^{|U|\times|T|\times I} \to \mathcal{P}(T \times \mathbb{Z}_I)$ which takes all DC-Net messages as input, and results a set of time slot-partial vote index-pairs (t, i) which should be disclosed, can be defined as

$$\mathcal{D}(\boldsymbol{d}) = \left\{ (t, i) : \sum_{u\in U} \bar{d}_{u,t,i} = \min_{t'\in T, i'\in \mathbb{Z}_I} \left\{ \sum_{u\in U} \bar{d}_{u,t',i'} \right\} \right\}. \tag{9}$$

However, as already discussed in Section 5.1, one may decide not to disclose availability patterns with the drawback, that attacks to the availability of the poll are possible then.

The privacy of a message in the DC-Net depends only on the secrecy of the keys. Also the privacy of the original scheme relied only on this secrecy. Splitting the vote vector into several parts introduces a new point of attack. Now, the anonymity of the message also depends on the randomness and secrecy of the partial vote index. If an attacker can predict at which DC-Net rounds messages from some participants occur, she can separate the other messages into smaller anonymity sets. If all participants distribute their votes randomly over all rounds, Mallory needs the cooperation of the other participants to deanonymize her victim. However, deanonymization can also be done without the help of the different partial DC-Nets, as participants may also disclose their shared DC-Net keys.

In a successful schedule with no attacker, one sum for every time slot with the number of all available participants is disclosed. This is more disclosure than demanded in the privacy requirement. However, the information which can be extracted from the sum of the available participants is still very low.

normal poll

	t_0	t_1	t_2	t_3
Alice	0	1	0	0
Bob	1	1	0	1
Mallory	0	0	0	2
$\sum$	1	2	0	3

inverted poll

	t_0	t_1	t_2	t_3
Alice	1	0	1	1
Bob	0	0	1	0
Mallory	1	1	1	−1
$\sum$	2	1	3	0

$\sum$	3	3	3	3

Figure 4: By the use of an inverted poll, the (+2)-attack can be reduced to the (−1)-attack. Mallory has to send a −1 at t_3 in the right table, because the sum of t_3 of both tables would not be equal to the number of participants otherwise.

6. PREVENTING (+2)-ATTACKS

In the example of Figure 1b, Alice can detect a (+2)-attack of Mallory as she knows that the sum is an element of $\{0, \ldots, |U| - 1\}$. However, as we do not request unanimous agreement anymore, it may be the case that Alice and Bob did not vote for a time slot (cp. e. g., time slot t_2 of Figure 1b) where Mallory sends a 2. In such a case neither Alice nor Bob can detect the attack on their own.

A simple solution to this attack would be to request the verification phase in any case for the agreed time slot. This would neither be privacy-friendly nor efficient in terms of message exchanges.

In the following, another solution is proposed. In addition to the normal poll, every participant votes for the same time slots in an inverted poll. Every participant u calculates for every time slot t an inverted vote $v'_{u,t}$ which depends on his vote $v_{u,t}$ such that

$$v_{u,t} + v'_{u,t} = 1. \tag{10}$$

With the inverted votes $v'_{u,t}$, an inverted poll is done in the same way the normal poll was done.

The sum of both result vectors, the one from the normal poll and the one from the inverted poll, should be a vector where all elements are equal to the number of participants $|U|$. By checking this property, it is ensured that every participant calculated the inverted vote vector according to Equation 10. If Mallory wants to send a 2 for some time slot, she then has to send a −1 in the inverted poll. However, splitting the inverted vote vector into several ones, this attack can be prevented in the same way (−1)-attacks were prevented within the vote vector.[10] Figure 4 illustrates the whole process.

Let $\bar{\boldsymbol{d}}$ be the 3-dimensional array of all DC-Net messages sent to the normal poll and $\bar{\boldsymbol{d}}'$ be the 3-dimensional array of all DC-Net messages sent to the inverted tables. The function

[10]Note that the index of the partial vote which contains the inverted vote is chosen independently from the partial vote index in the normal poll.

which checks the correctness of the inverted vote calculation is defined as

$$\mathcal{C}(\bar{\boldsymbol{d}}, \bar{\boldsymbol{d}}') = \begin{cases} \text{true} & \text{if } \forall t \in T: \\ & |U| = \sum_{u \in U, i \in \mathbb{Z}_I} \left(\bar{d}_{u,t,i} + \bar{d}'_{u,t,i} \right) \\ \text{false} & \text{otherwise.} \end{cases} \quad (11)$$

6.1 Verifiability

When evaluating the result, two kinds of inconsistencies may occur: the sum of both polls may be lower or higher than the number of participants. As we split the votes into several ones (cp. Section 5), we do not consider (-1)-attacks at this point. Therefore, a value lower than the number of participants may occur only if one or more participants sent $v_{u,t} + v'_{u,t} = 0$. Having such a case would mean that the number of available participants σ_t would be the result of the normal poll. The difference of the number of available participants and the total number of participants are wrongly cast votes.

The second kind of inconsistency is if the sum of both polls is higher than the number of participants. As we prevented (-1)-votes, the result of the normal poll at a time slot t should be greater or equal than the number of available participants at this time slot. In addition, the number of available participants is greater than the total number of participants minus the result of the inverted poll at a certain time slot. Putting both inequations together, one obtains a range which expresses the possible number of available participants:

$$\sum_{i \in I, u \in U} \bar{d}_{u,t,i} \geq \sigma_t \geq |U| - \sum_{i \in I, u \in U} \bar{d}'_{u,t,i}. \quad (12)$$

However, as an attacker may manipulate his vote in a way that this range results in $\{0, \ldots, |U|\}$, he may attack the availability of the poll in such a way. To unmask the attacker, all DC-Net rounds for the inconsistent time slot can be decrypted as shown before. If the attack discovery goes along with some cost (penalty, reputation loss, etc.), it makes such attacks unattractive.

6.2 Privacy

During the verification phase of an inconsistent inverted poll, the availability of all participants at the inconsistent time slot are disclosed. To avoid disclosure of all availabilities, one may disclose the DC-Net rounds step by step and stop when the attacker is found. The sequence of disclosure should therefore be a fixed order, which is not known before every participant stated his vote.[11]

In a successful run, no more information is disclosed than in the old scheme, the inverted poll contains only redundant information.

6.3 Untrusted Single Entity

The central server acts only as a blackboard which has to publish all messages after everybody has cast his vote. If the server publishes votes before vote casting is complete, an attacker may calculate the result before submitting his vote. To avoid this restriction, one additional communication phase would be needed in which all participants commit to their vote. Compared to the old scheme, the same trust assumptions are put into the server.

[11]E. g., every participant may commit himself to a random number together with his vote vector. In case of verification all commitments are revealed and the random numbers are added to one single seed which is used to bootstrap a sequence.

Table 3: Comparison of the computational complexity of the original scheme with unanimous agreement and the scheme with our extension. In both cases we assume no attack appearing

	unanimous agreement	new scheme
discrete exp.	$\|U\| - 1$	$\|U\| - 1$
symmetric decr.	$\|T\| \cdot (\|U\| - 1)$	$2 \cdot I \cdot \|T\| \cdot (\|U\| - 1)$
hashes	$\|T\| \cdot (\|U\| - 1)$	$2 \cdot I \cdot \|T\| \cdot (\|U\| - 1)$

6.4 Usability

In a successful poll, two communication phases are needed. The first one is for casting the votes and the second one is for result publication Compared to a poll in another scenario (e. g., Doodle), the same number of message exchanges and user interactions are needed in a successful run. However, every participant has to register at a central server once to setup the asymmetric key pair.

Our extension has no negative influence to the number of message exchanges compared to the old scheme therefore the same number of user interactions is needed.

6.5 Efficiency

The extension presented in this paper affects the computational complexity of the old scheme only in terms of symmetric cryptographic operations, i. e., the amount of asymmetric cryptographic operations is not affected. To be precise, in the old scheme without our extension every participant has to calculate $|T| \cdot (|U| - 1)$ symmetric decryptions and hashes in a successful poll. Assuming I simultaneous DC-Net rounds per time slot in our extension for (-1)-attack prevention and a successful run (no attacks occurred), every participant has to calculate $I \cdot (|U| - 1)$ decryptions and hashes. Using an inverted poll doubles the number symmetric operations. Table 3 compares the efficiency of our extension with the efficiency of the original scheme.

7. IMPLEMENTATION

The protocol has already been implemented [22]. The cryptographic operations are implemented in JavaScript; no installation on client side is needed. The participant has to trust the server that the JavaScript is delivered correctly. To increase the acceptance even more, it is possible to vote without non-anonymous (the Doodle way) and anonymous in the same poll. Two screenshots of the application are shown in Figure 5. In Figure 5a, one can see the status shortly before Bob casts his vote. There, the tooltip shows the 8-digit hex id of Mallory's Diffie–Hellman key.

Figure 5b shows how a cheater is detected. Mallory tried to send a -1 at the second time slot and a $+2$ at the forth one

Table 4: Performance measurement of the key calculation in a poll with $|U| = 5$, $I = 20$ and $|T| = 10$ on an Intel Pentium 4 Duo with 2.8 GHz, 2 GB RAM running Windows XP SP3

	AES256+SHA256	DH	total
Internet Explorer 8	9.4 s	15.3 s	28.9 s
Firefox 3.6.6	9.1 s	7.4 s	18.7 s
Safari 5.0	1.4 s	8.9 s	12.9 s
Opera 10.60	0.8 s	2.0 s	3.7 s
Google Chrome 5.0	1.3 s	2.5 s	5.1 s

($\mathcal{V}(\bar{\boldsymbol{d}})$ failed). At the third time slot she sent inconsistent values for the normal and inverted poll and therefore $\mathcal{C}(\bar{\boldsymbol{d}}, \bar{\boldsymbol{d}}')$ failed.

The implementation has been done using the JavaScript BigInteger library from Tom Wu [40]. For the symmetric cipher, AES with 128 bit key length is used. SHA256 is used for the hash function. For AES and SHA, the JavaScript libraries from B. Poettering are used [35].

Table 4 shows a performance measurement of the key calculation for an example poll. One can see, that the calculation needs about 20 s, using Firefox. From the first point of view, this looks if the application is unusable. When browsers run a script which needs longer calculation time, it is usual that they ask the user if he wants to stop it. To avoid these pop-ups, the BigInteger library was modified in a way that it calculates exponentiations asynchronously with a callback function. This enables that the calculation is forked in the background and the browser can operate which avoids nasty pop-ups. Additionally, the user can enter his preferences, while the browser calculates the keys. After the calculation has been done, the submit button is enabled, and the participant can submit his vote. Assuming, that a user needs about 20 s to enter his preferences (look up the time slots in his personal calendar, click the buttons etc.), there is no extra time to wait for.

To enhance the performance when users participate in several polls or request some poll more than once, the DOM storage is used [19]. Already calculated values are stored there to speed up the calculations later on. This storage may be target of possible XSS-attacks of course, but this is out of scope for this paper.

8. CONCLUSION

We proposed a scheme, which is able to schedule events in a privacy friendly way. Furthermore we implemented the scheme in a Web 2.0 application. Due to the use of JavaScript for all client side operations, no installation is needed for the user. We therefore showed, that complex cryptography is possible in zero footprint applications. Non-anonymous votes as well as anonymous ones can be cast in the same poll, which increases the acceptance of the application.

As a next step, it should be investigated how other applications can be made privacy-friendly without negatively influencing the usability. Instead of trying to solve huge identity management solutions, one should think more of small applications where privacy-preserving features are feasible.

As small mobile devices are becoming more and more important, they should be the target of privacy-friendly applications research as well. Special problems arise here due to the restriction of computation power, limited energy, and the small user interface.

For our application, we already have some more extensions in mind which should be investigated. Furthermore we want to investigate whether users understand the underlying concepts (e. g., the unusual concept, of identifying themselves to gain more privacy, or that they have to give up privacy to unmask attackers in some cases).

References

[1] M. Abe. Universally verifiable mix-net with verification work indendent of the number of mix-servers. In *EUROCRYPT*, pages 437–447. Springer, 1998.

[2] B. Adida. Helios: Web-based open-audit voting. In P. C. van Oorschot, editor, *USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.

[3] B. Adida. Helios voting. `http://v1.heliosvoting.org/`, July 2010.

[4] L. Baird. BigIntegers in JavaScript. `http://www.leemon.com/crypto/BigInt.html`, July 2010. Version 5.4.

[5] O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard. Practical multi-candidate election system. In *PODC '01: Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, pages 274–283, New York, NY, USA, 2001. ACM.

[6] J. C. Benaloh and M. Yung. Distributing the power of a government to enhance the privacy of voters. In *PODC '86: Proceedings of the fifth annual ACM symposium on Principles of distributed computing*, pages 52–62, New York, NY, USA, 1986. ACM.

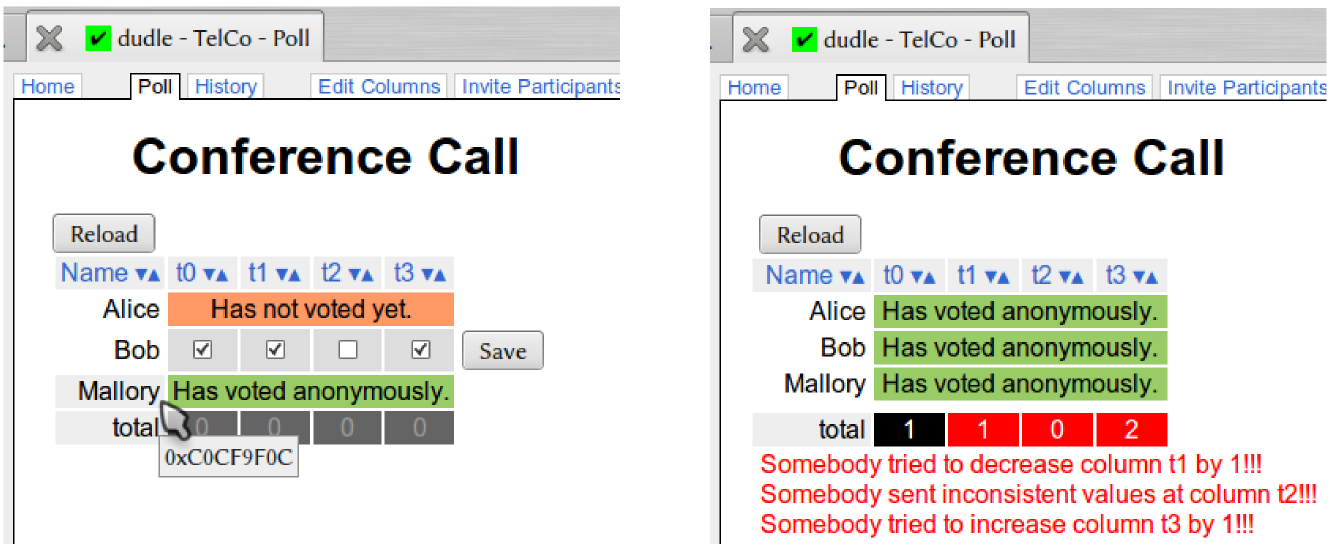
[7] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, Jan. 1988.

[8] D. Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In *Lecture Notes in Computer Science on Advances in Cryptology-EUROCRYPT'88*, pages 177–182, New York, NY, USA, 1988. Springer-Verlag New York, Inc.

[9] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.

[10] M. R. Clarkson, S. Chong, A. C. Myers, M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. In *IEEE Symposium on Security and Privacy*, pages 354–368. IEEE Computer Society, 2008.

[11] J. D. Cohen and M. J. Fischer. A robust and verifiable cryptographically secure election scheme. In *SFCS '85: Proceedings of the 26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pages 372–382, Washington, DC, USA, 1985. IEEE Computer Society.



Conference Call

Name	t0	t1	t2	t3
Alice	Has not voted yet.			
Bob	☑	☑	☐	☑
Mallory	Has voted anonymously.			
total	0	0	0	0

0xC0CF9F0C

Conference Call

Name	t0	t1	t2	t3
Alice	Has voted anonymously.			
Bob	Has voted anonymously.			
Mallory	Has voted anonymously.			
total	1	1	0	2

Somebody tried to decrease column t1 by 1!!!
Somebody sent inconsistent values at column t2!!!
Somebody tried to increase column t3 by 1!!!

(a) A screenshot of the poll, before vote casting. At the tooltip, one can see the id of Mallory's key.

(b) Mallory tried to cheat but her malicious vote can be detected.

Figure 5: Screenshots of an example poll.

[12] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[13] B. W. DuRette. Multiple administrators for electronic voting. Bachelor's thesis, Massachusetts Institute of Technology, May 1999.

[14] M. S. Franzin, E. C. Freuder, F. Rossi, and R. Wallace. Multi-agent meeting scheduling with preferences: Efficiency, privacy loss, and solution quality. AAAI Technical Report WS-02-13, 2002.

[15] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In J. Seberry and Y. Zheng, editors, *AUSCRYPT*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer, 1992.

[16] R. Greenstadt, J. P. Pearce, E. Bowring, and M. Tambe. Experimental analysis of privacy loss in DCOP algorithms. In *Proc. of ACM AAMAS*, pages 1424–1426, New York, 2006. ACM Press.

[17] T. Herlea, J. Claessens, B. Preneel, G. Neven, F. Piessens, and B. D. Decker. On securely scheduling a meeting. In M. Dupuy and P. Paradinas, editors, *SEC*, volume 193 of *IFIP Conference Proceedings*, pages 183–198. Kluwer, 2001.

[18] M. A. Herschberg. Secure electronic voting over the world wide web. Master's thesis, Massachusetts Institute of Technology, May 1997.

[19] I. Hickson. Web storage. Last call WD, W3C, Dec. 2009. http://www.w3.org/TR/2009/WD-webstorage-20091222/.

[20] IEEE/IFIP. *Proceedings of IEEE International Conference on Computational Science and Engineering*, volume 3, Los Alamitos, CA, USA, Aug. 2009. IEEE Computer Society. Conference on Information Privacy, Security, Risk and Trust (PASSAT '09).

[21] M. Jakobsson. A practical mix. In *EUROCRYPT*, pages 448–461. Springer, 1998.

[22] B. Kellermann. Dudle homepage. `http://dudle.inf.tu-dresden.de`, July 2010.

[23] B. Kellermann. Open research questions of privacy-enhanced event scheduling. In J. Camenisch and V. Kisimov, editors, *Proceedings of iNetSec 2010: Open Research Problems in Network Security*, Lecture Notes in Computer Science. Springer, 2010. to appear.

[24] B. Kellermann and R. Böhme. Privacy-enhanced event scheduling. In *Proceedings of IEEE International Conference on Computational Science and Engineering* [20], pages 52–59. Conference on Information Privacy, Security, Risk and Trust (PASSAT '09).

[25] T. Léauté and B. Faltings. Privacy-preserving multi-agent constraint satisfaction. In *CSE (3)* [20], pages 17–25. Conference on Information Privacy, Security, Risk and Trust (PASSAT '09).

[26] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed multi-event scheduling. In *AAMAS*, pages 310–317. IEEE Computer Society, 2004.

[27] R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent*

Systems, pages 438–445, Washington, DC, USA, 2004. IEEE Computer Society.

[28] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161:149–180, 2004.

[29] M. Näf. Doodle blog. `http://www.doodle.com/blog/2010/06/24/`, June 2010.

[30] M. Näf. Doodle homepage. `http://www.doodle.com`, July 2010.

[31] W. Ogata, K. Kurosawa, K. Sako, and K. Takatani. Fault tolerant anonymous channel. In Y. Han, T. Okamoto, and S. Qing, editors, *ICICS*, volume 1334 of *Lecture Notes in Computer Science*, pages 440–444. Springer, 1997.

[32] M. Ohkubo, F. Miura, M. Abe, A. Fujioka, and T. Okamoto. An improvement on a practical secret voting scheme. In M. Mambo and Y. Zheng, editors, *ISW*, volume 1729 of *Lecture Notes in Computer Science*, pages 225–234. Springer, 1999.

[33] C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *EUROCRYPT*, pages 248–259, 1993.

[34] A. Pfitzmann and M. Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. `http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.33.pdf`, Apr. 2010. v0.33.

[35] B. Poettering. The AES block cipher and the SHA256 message digest in JavaScript. `http://point-at-infinity.org/`, July 2010. Version 0.1.

[36] K. Sako. Electronic voting scheme allowing open objection to the tally. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 77(1):24–30, 1994.

[37] K. Sako and J. Kilian. Secure voting using partially compatible homomorphisms. In *CRYPTO '94: Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, pages 411–424, London, UK, 1994. Springer-Verlag.

[38] D. Shapiro. BigInt, a suite of routines for performing multiple-precision arithmetic in JavaScript. `http://ohdave.com/rsa/BigInt.js`, July 2010.

[39] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Asynchronous search with aggregations. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 917–922. AAAI Press / The MIT Press, 2000.

[40] T. Wu. BigIntegers and RSA in JavaScript. `http://www-cs-students.stanford.edu/~tjw/jsbn/`, July 2010.

[41] M. Yokoo and K. Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, 2000.

APPENDIX

A. COMPLETE MAJORITY AGREEMENT PROTOCOL

The complete protocol, consists of three mandatory phases (initialization, vote casting, and result publication/verification) and one optional attacker identification phase that is run when inconsistencies occur. We will shortly describe the phases in the following. Before the protocol can be run, it is necessary that every participant registers at the poll server or exchange a public key with everybody else. This registration is done once, the key can be used to every following poll.

A.1 Registration

Let q be the modulus and g the generator of the Diffie–Hellman key agreement protocol, both are constant for all potential participants and polls. Each participant u registers in three steps:

1. Fetch the modulus q and the generator g from the server.
2. Choose a random number and store it as Diffie-Hellman secret key sec_u.
3. Calculate the public key $\mathsf{pub}_u = g^{\mathsf{sec}_u} \bmod q$ and publish it on the server.

A.2 Initialization

The initialization phase is quite similar to a Doodle poll [30]. The initiator defines a set of time slots T and an ordered set of participants U. In difference to a Doodle poll, the set of participants U is fixed from the beginning and each participant has to know this set.[12] However, dynamic inclusion and exclusion of participants were already discussed. [24, Sects. V-C, V-D].

All parameters are sent to all participants.

A.3 Vote Casting

In the vote casting phase, every participant has to state a vote vectors, of size $|T|$, which contains elements $v_{u,t} \in \{0,1\}$. For each element $v_{u,t}$, 0 means that the participant u is unavailable at time slot t, 1 signals availability.

Every participant u calculates for every time slot t an inverted vote $v'_{u,t} = v^1_{u,t}$ which depends on his vote $v_{u,t} = v^0_{u,t}$ such that

$$v^0_{u,t} + v^1_{u,t} = 1. \qquad (13)$$

Let $I \in \mathbf{N}$ be a security parameter.[13] Every participant splits for every time slot $t \in T$ and $x \in \{0,1\}$, each element $v^x_{u,t}$ of his vote vector into I partial votes such that:

1. An index $j \in \mathbb{Z}_I$ for one partial vote, is chosen randomly and kept secret.
2. The partial vote with index j ($\bar{v}^x_{u,t,j}$) is equal to the participants actual vote $v^x_{u,t}$.

[12]The set of participants acts additionally as anonymity set.
[13]compare Table 2

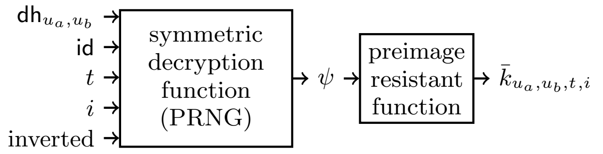


Figure 6: Generation of a DC-Net key.

3. The remaining $I-1$ partial votes are equal to 0.

Every participant u_a calculates for every other participant $u_b \in U, u_a \neq u_b$ a Diffie–Hellman secret

$$\mathsf{dh}_{u_a,u_b} = g^{\mathsf{sec}_{u_a} \cdot \mathsf{sec}_{u_b}} \bmod q. \tag{14}$$

For every time slot $t \in T$ and $x \in \{0,1\}$, I DC-Net keys $\bar{k}^x_{u_a,u_b,t,i}$ are generated. Let id be a universal unique poll identifier.[14] Let $\mathrm{decr}_{\mathsf{key}}(\mathsf{ciphertext})$ be a decryption function, which resists adaptively chosen plain-cipher-text attacks. Let n be a modulo for the DC-Net, and $\mathrm{h}(\cdot) \bmod n$ a preimage resistant hash function. Every participant u_a generates DC-Net keys with another participant u_b

$$\bar{k}^x_{u_a,u_b,t,i} = \begin{cases} \mathrm{h}(\mathrm{decr}_{\mathsf{dh}_{u_a,u_b}}(\mathsf{id}||t||i||x)) \bmod n & \text{if } u_a < u_b \\ -\mathrm{h}(\mathrm{decr}_{\mathsf{dh}_{u_a,u_b}}(\mathsf{id}||t||i||x)) \bmod n & \text{otherwise.} \end{cases} \tag{15}$$

Figure 6 illustrates the key generation process.

After the key calculation has been done, every participant u adds his keys to the elements of his vote vectors:

$$\bar{d}^x_{u,t,i} = \bar{v}^x_{u,t,i} + \sum_{u' \in U, u \neq u'} \bar{k}^x_{u,u',t,i} \tag{16}$$

All encrypted votes are sent to the central server and published after all participants have cast their votes.

A.4 Result Publication and Verification

When all encrypted votes are published, every participant can calculate the result by adding all encrypted votes of one time slot. The result for time slot t is calculated by

$$\sigma_t = \sum_{i \in \mathbb{Z}_I, u \in U} \bar{d}_{u,t,i} \tag{17}$$

To verify the result, every participant u_p makes 3 checks:

1. $\forall t \in T, i \in \mathbb{Z}_I, x \in \{0,1\} : \sum_{u \in U} \bar{d}^x_{u,t,i} \in \{0,\dots,|U|\}$

2. $\forall (t,i,x) \in \{(t,i,x) : t \in T, i \in \mathbb{Z}_I, x \in \{0,1\}, \bar{v}^x_{u_p,t,i} = 1\} : \sum_{u \in U} \bar{d}^x_{u,t,j} > 0$

3. $\forall t \in T : |U| = \sum_{u \in U, i \in \mathbb{Z}_I} \bar{d}^0_{u,t,i} + \bar{d}^1_{u,t,i}$

[14]The poll identifier avoids generation of similar shared keys for different polls, which use the same time slot.

A.5 Optional Attacker Identification

If one of the checks, stated in the previous section fails, the attacker should be unmasked. Depending on the check which failed, the participant has to decide, what to do:

1. The DC-Net round where $\sum_{u \in U} \bar{d}^x_{u,t,i} \notin \{0,\dots,|U|\}$ is decrypted.

2. The participant u has to decide if he wants to give up his privacy and asks to decrypt the DC-net round where $\bar{v}^x_{u,t,i} = 1$ and $\sum_{u \in U} \bar{d}^x_{u,t,j} \leq 0$. This would reveal his availability at this time slot but unmask the attacker.

3. All votes for time slot t are decrypted where $|U| \neq \sum_{u \in U, i \in \mathbb{Z}_I} \bar{d}^0_{u,t,i} + \bar{d}^1_{u,t,i}$. This can be done as described in Section 6.2.

If keys between participants u_a and u_b have to be released, $\mathrm{decr}_{\mathsf{dh}_{u_a,u_b}}(\mathsf{id}||t||i||x)$ is released instead. The preimage resistant hash function prevents cheating with keys here.