Privacy-Enhanced Web-Based Event Scheduling with Majority Agreement

Complete Protocol

Benjamin Kellermann

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

1 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.

1.1 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.

1.2 Initialization

The initialization phase is quite similar to a Doodle poll [1]. 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.[1] However, dynamic inclusion and exclusion of participants were already discussed. [2, Sects. V-C, V-D].

All parameters are sent to all participants.

[1] The set of participants acts additionally as anonymity set.

1.3 Vote Casting

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

Every participant u calculates for every time slot t a check vote $v^1_{u,t}$ which depends on his vote $v^0_{u,t}$ such that

$$v^0_{u,t} + v^1_{u,t} = 1. \tag{1}$$

Let $I \in \mathbf{N}$ be a security parameter. 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 $v^x_{u,t}$.
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

$$\mathsf{dh}_{u_a,u_b} = g^{\mathsf{sec}_{u_a} \cdot \mathsf{sec}_{u_b}} \bmod q. \tag{2}$$

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.[2] 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

$$\bar{k}^x_{u_a,u_b,t,i} = \begin{cases} \mathrm{h}(\mathrm{decr}_{\mathsf{dh}_{u_a,u_b}}(\mathsf{id}||t||i||x)) \bmod n & \text{if } u_a < u_b \\ -\mathrm{h}(\mathrm{decr}_{\mathsf{dh}_{u_a,u_b}}(\mathsf{id}||t||i||x)) \bmod n & \text{otherwise.} \end{cases} \tag{3}$$

Fig. 1 illustrates the key generation process.

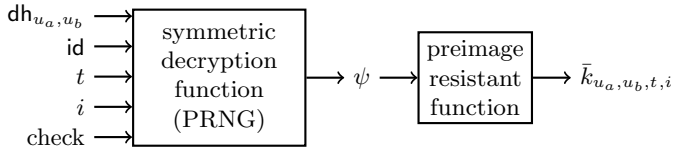

Fig. 1. Generation of a DC-Net key.

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

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

$$\bar{d}^x_{u,t,i} = \bar{v}^x_{u,t,i} + \sum_{u' \in U, u \neq u'} \bar{k}^x_{u,u',t,i} \tag{4}$$

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

1.4 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

$$\sigma_t = \sum_{i \in \mathbb{Z}_I, u \in U} \bar{d}_{u,t,i} \tag{5}$$

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

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

1.5 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 $\sum_{u \in U} \bar{d}^x_{u,t,i} \notin \{0, \ldots, |U|\}$ 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 $|U| \neq \sum_{u \in U, i \in \mathbb{Z}_I} \bar{d}^0_{u,t,i} + \bar{d}^1_{u,t,i}$.

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.

References

1. Näf, M.: Doodle: easy scheduling. `http://www.doodle.com` (November 2010)
2. Kellermann, B., Böhme, R.: Privacy-enhanced event scheduling. In: PASSAT '09. Volume 3., Los Alamitos, CA, USA, IEEE/IFIP, IEEE Computer Society (August 2009) 52–59