Privacy-Preserved Meeting Organization

Group 06

January 14, 2025

1 Basic definitions

Following finite sets are defined:

- $\mathcal{D}$: The set of all documents.
- $\mathcal{R}$: The set of all roles.
- $\mathcal{I}$: The set of all individuals
- $\mathcal{L}$: The set of all locations.
- $\mathcal{T}$: The set of all time slots.

2 Access Control List

We define following relationships, using above definitions.

$$d = \{d \in \mathcal{D} \mid \text{d is a document}\}$$

$$i = \{i \in \mathcal{I} \mid i \text{ is an individual }\}$$

$$g = \{g \subseteq \mathcal{I} \mid g \text{ is a subset of one or more individuals in } \mathcal{I}\}$$

Above relationships mean that d is an element of set $\mathcal{D}$, and i is an element of set $\mathcal{I}$. Further, g is a group of one or more individuals (i), where $i \in \mathcal{I}$, such that $g \neq \emptyset$.

Consider that following finite set is also defined:

- $\mathcal{G}$: Set of all possible not-null subsets of $\mathcal{I}$

Based on above all sets, we define following relationship.

$$access(d) = \{g \in \mathcal{G} \mid g \text{ has access to } d\}$$

Above relationship means that g is an element of set $\mathcal{G}$, and that $access(d)$ is

the set of groups (g) having access permission to document d.
Here we note that, $access(d) = \mathcal{G}$ converts d to a **public document**.

By above last two relationships, since any element g of $access(d)$ is also a subset of $\mathcal{I}$, such that $g \subseteq \mathcal{I}$, we have the relationship $access(d) \subseteq \mathcal{I}$, when $access(d)$ is defined in form of **singleton subsets** of $\mathcal{I}$. It implies also that $|access(d)| \leq |\mathcal{I}|$, when $access(d)$ is defined in the form of singleton subsets of $\mathcal{I}$. Simply, a singleton subset of $\mathcal{I}$ includes an individual (i).
Regarding that inequality, $|access(d)| = |\mathcal{I}|$ is the situation when every i in $\mathcal{I}$ is present in at least one group (g), such that $g \subseteq access(d)$. At such a situation, both relationships $access(d) = \mathcal{G}$ and $|access(d)| = |\mathcal{I}|$ imply the same meaning that, document is a **public** document.

3 Meeting agenda

Agenda of a meeting is the document that defines the set of groups (g) required to attend the meeting, where **group** has same meaning as defined above. When we consider agenda as document d, those groups (g) are elements of set $access(agenda)$.

Theoretically it is possible to require all individuals of set $\mathcal{I}$ or all available not-null subsets of set $\mathcal{I}$, to attend a single meeting. But, in practical scenario, probability of organizing such a meeting is low.

However, there are both private meetings and public meetings, in our scope. If $access(d) = \mathcal{G}$ is used for meeting agenda of public meetings, it's impossible to distinguish the intended participant groups explicitly. Therefore, in agenda document of public meetings, we include a group labeled as **public** group, in addition to the actual intended participant groups of meeting, to state that agenda is **public**. So, on the other hand, absence of group labeled as **public** in $access(agenda)$ means that, meeting is **private**.

- If there is at least one document in meeting, such that $access(d) \neq \mathcal{G}$, **public** group shouldn't have access to meeting agenda.
- If every documents in meeting has $access(d)$ such that $access(d) = \mathcal{G}$, **public** group can have access to meeting agenda.
- If agenda is the only document in meeting, **public** label can be used by meeting organizer to define whether agenda document is **private** or **public** (i.e. whether meeting is private or public).

Following flow chart depicts the process of identifying whether a document is **private** or **public**.

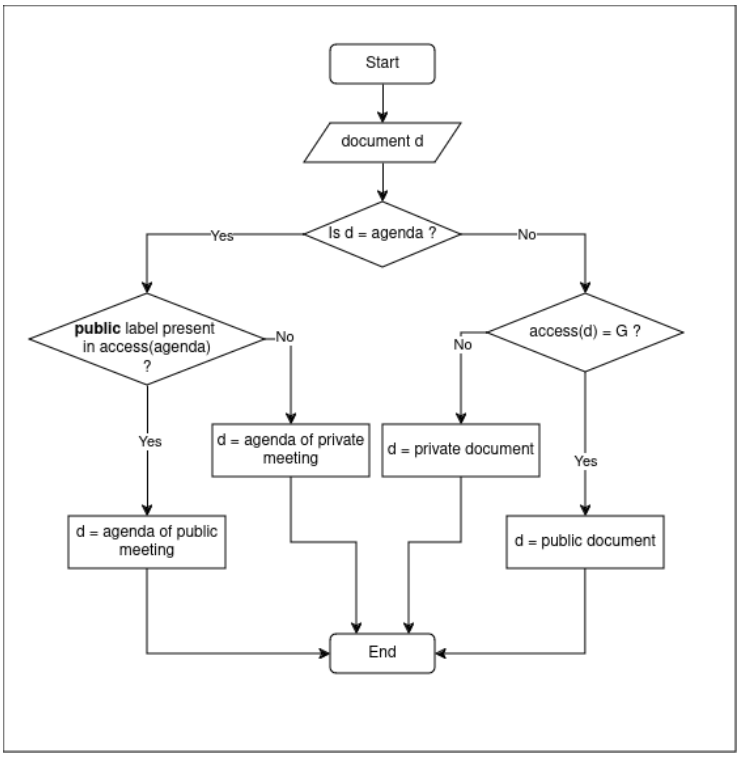

Figure 1: Process to identify whether a document is private or public

Regarding other documents except agenda, for simplicity of implementation, we can include only **public** label in access control list, without including any other group of $\mathcal{G}$, to mean that $access(d) = \mathcal{G}$ or that document is a public document. Because, we do not need to include any other group (g) in a public non-agenda document, though we required them in public agenda documents for identifying meeting participants.

In addition to presence or absence of **public** label in $access(agenda)$, meeting agenda should define the **meeting quorum**, for the meeting. This theme will be discussed later.

4 Definition of a meeting

We assume that every meeting has an agenda associated with it, to define the set of groups(g) required to attend the meeting. Agenda of a particular meeting M is a document, belonging to set $\mathcal{D}$.

When we consider the agenda document of meeting M, for every group g invited

to meeting M; $g \in access(agenda)$. Also consider that, D represents set of documents discussed in M, including agenda, such that $D \subseteq \mathcal{D}$. Hence, according to the assumption emphasized above, for any meeting M; $|D| \geq 1$.

For conducting a meeting, at least 2 individuals are required. Consider that I represents the set of individuals attending meeting M, such that $I \subseteq \mathcal{I}$, when groups (g) of $access(agenda)$ are converted to corresponding elementary individuals (i). Here we note that, for any meeting M; $|I| \geq 2$.
Accordingly, when $access(agenda)$ is defined in terms of singleton subsets of $\mathcal{G}$, and those groups (singleton subsets) in $access(agenda)$ are represented by G, such that $G \subseteq \mathcal{G}$, it can be observed that $|G| \geq 2$.

Consider set of locations of individuals in M as L (in other words, set of locations of indiduals in set I, during meeting time), such that $L \subseteq \mathcal{L}$. Every individual attends meeting from a particular location l, such that $l \in L$. We note that, if meeting is online or hybrid, $|L| > 1$. If meeting is onsite, $|L| = 1$, since every individual is at same location. Every meeting should be in one mode, out of online, hybrid, onsite modes. Therefore, for any meeting M; $|L| \geq 1$.

Since a **meeting** is a **synchrnous** communication, every individual in meeting M should attend the meeting during the same time slot t (Assuming that all individuals are in same time zone).

Based on these definitions, we define meeting M as a 4-tupple,

$$M =< D, I, L, t >$$

such that,

$$D \subseteq \mathcal{D}$$

$$L \subseteq \mathcal{L}$$

$$I \subseteq \mathcal{I}$$

$$t \in \mathcal{T}$$

5 Transformation of individual into role

Consider that same sets defined above will be used in explanations below, in same notations:

Consider g and g' as subsets of $\mathcal{G}$ such that $g, g' \subseteq \mathcal{G}$. And consider d as a **private** document , l as a location and t as a time slot such that $d \in \mathcal{D}$, $l \in \mathcal{L}$ and $t \in \mathcal{T}$. Further consider that $g \in access(d)$ and $g' \notin access(d)$, for restricting access of document d, where $|access(d)| = n$, given that $access(d)$ is defined as a set of singleton subsets of $\mathcal{G}$. A singleton subset of $\mathcal{G}$ means an elementary subset g, in which only one element (i.e. only one individual i) is present.

Also note that, i and i' are two individuals representing subsets g and g', respectively.

Assume that at scenario 1, i attends a **meeting** at location l during time slot t to discuss document d, where i' has no access to location l during same time slot t.
Here we state that privacy of meeting discussing document d was preserved at context $l \times t$

Now assume that at scenario 2, i attends a **meeting** at location l during time slot t to discuss document d, where i' also has access to location l during same time slot t.
Here we state that privacy of meeting discussing document d was violated at context $l \times t$, because $n+1$ individuals including i' have got access to content of document d. But actually $|access(d)| = n$, when $access(d)$ is defined as a set of singleton subsets of $\mathcal{G}$, as mentioned above. We observe that $(n+1) \geq |access(d)| = n$

When above 2 scenarios are compared, we observe that role of same individual i, representing subset g such that $g \in access(d)$, has experienced a variation. Context of i has changed, depending on location and time.

Therefore we define that presence of i at context $l \times t$ transforms i to role r.

$$transform(i, l, t) = r : r \text{ is role of } i \text{ at location } l \text{ at time slot } t$$

If $g \in access(d)$, $g' \notin access(d)$ and d is a **private** document, i representing g should attend a meeting to discuss d at context $l \times t$, only if i' representing g' has no access to $l \times t$. Accordingly, to identify the privacy preserving context for discussing document d, combination of i, l, t is required.

6 Difference between public and private roles

When we consider a **private** document d, we can't exactly predict the time, at which i', representing g', such that $g' \notin access(d)$, will get access to location l. Therefore, meeting organizer has the responsibility of defining location l as a **private** location or a **public** location, considering whether access of i' has been strictly restricted, during all potential meeting time slots (represented by set $\mathcal{T}$).

Using this definition and above formula, we can show that, i representing g, such that $g \in access(d)$ where d is a **private** document, is transformed to role g itself, at a **private** location. Here, location should be defined as a **private** location, by same entity, that defined the set $access(d)$ for document d.

$$transform(i, l, t) = r$$

$$transform(i, (private\_location), t) = r$$
$$transform(i, (private\_location), t) = g$$

On the other hand, any location l is defined as a **public** location, if access of i' has **not** been strictly restricted, during any potential meeting time slot in set of time slots $\mathcal{T}$.

Using this definition and above formula, we can show that, i representing g, such that $g \in access(d)$, is transformed to **public** role, at a **public** location. Location should be defined as a **public** location, by same entity, that defined the set $access(d)$ for document d.

$$transform(i, l, t) = r$$
$$transform(i, (public\_location), t) = r$$
$$transform(i, (public\_location), t) = public$$

Based on these derivations, we have identified a constraint relevant to i, for discussing d in a privacy preserved meeting.

Constraint: When d is a **private** document, every i representing g, such that $g \in access(d)$, that attends a meeting to discuss document d, must represent role g in the meeting.

When d is a **public** document, every i that attends a meeting to discuss document d, is allowed to represent **public** role in the meeting.

6.1 Roles in meeting agenda

If meeting agenda document does not include the group labeled as **public** in $access(agenda)$, it means that **public** $\notin access(agenda)$. Then i' representing g' such that $g' \notin access(d)$, should be strictly prevented from accessing the meeting, by conducting meeting at a **private** location, defined by relevant meeting organizing entity.

On the other hand, if meeting agenda includes group labeled as **public** in $access(agenda)$, it means that **public** $\in access(agenda)$. Then it is **not** mandatory to prevent access of i' representing g' such that $g' \notin access(d)$, for the meeting. Therefore, meeting can be conducted at a **private** location or **public** location, based on locations defined by relevant meeting organizing entity.

7 Variation of role

Now consider a situation where individual i representing g, such that $g \in access(d)$ has x number of locations, out of which any one can be selected for attending a meeting to discuss d. And assume that i has y number of time slots, out of which any one can be selected for attending the meeting.

We can depict the possible variations of $transform(i, l, t)$ function as below, for individual i, depending on locations defined by the entity, assuming that i doesn't change location during middle of a time slot.

(i)	t_1	t_2	...	t_{y-1}	t_y
l_1	x	x		x	x
l_2	x	x		x	x
...					
l_{x-1}	x	x		x	x
l_x	x	x		x	x

Table 1: Possibilities in variation of $transform(i, l, t)$ for individual i

Note that l_x represents the x^{th} location, while t_y represents the y^{th} time slot. Meanwhile x represents the role of i at the corresponding l and t (based on formula $transform(i, l, t) = r$). According to this representation, we observe that i has $x \times y$ number of possibilities at maximum, to attain the role.

Here we emphasize that some x roles can be categorized as **public**, with respect to **public** locations defined by an entity. According to the constraint identified, if d is a **private** document, i should attend the meeting only when $r = g$, such that $g \in access(d)$. When r = **public** role, individual i should strictly avoid discussing **private** documents. By following this constraint, access of i' representing g', such that $g' \notin aceess(d)$, into this meeting can be prevented.

8 Privacy of documents

8.1 Participants in access control lists of non-agenda documents

It is possible to discuss one or more documents in a meeting. Further, there can be both public documents and private documents among these documents. We do not need to follow any constraint to protect the privacy of public documents.

But when private documents are considered, it is needed to follow some con-

straints to pretect the privacy. For example, consider d_1 and d_2 as 2 private documents. An individual i representing r, such that $r = g$ and $g \in access(d_1)$, can be absent in access control list of d_2. In other words, $g \notin access(d_2)$ relationship can exist.

In this situation, discussing both d_1 and d_2 in same meeting can violate the privacy of d_2, when above mentioned indivisual i participates in that meeting. It means that, for discussing both d_1 and d_2 in same meeting, roles of all meeting participants should mandatorily be present in both $access(d_1)$ and $access(d_2)$. This relationship is graphically depicted in diagram below.

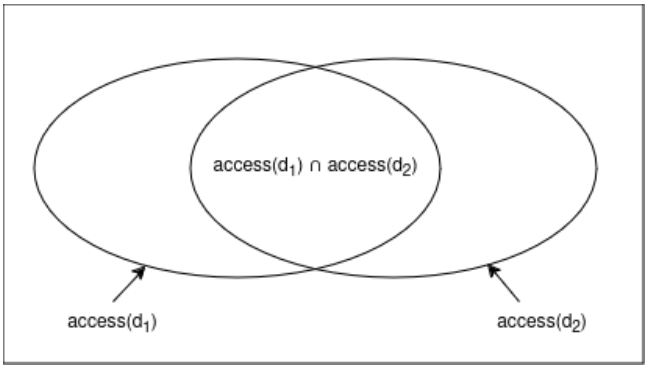

Figure 2: Intersection of access control lists of 2 private documents

This concept is applicable not only for 2 private documents, but also for any number of private documents discussed in same meeting. When there are n number of private documents to discuss in a meeting, intersection of access control lists of all those documents should be considered. Any individual i such that $i \notin (access(d_1) \cap access(d_2) \cap ... \cap access(d_{n-1}) \cap access(d_n))$ should be prevented from accessing the meeting.

8.2 Meeting participant validation

When there are private documents to discuss in a meeting, it is required to validate intersection of participants identified as above, with participants included in access control list of meeting agenda (see "Meeting agenda" section for more details on meeting agenda).
For protecting privacy of n number of private documents defined as $d_1, ..., d_n$, following relationship must be satisfied for the meeting.

$$access(agenda) \subseteq \{access(d_1) \cap access(d_2) \cap ... \cap access(d_{n-1}) \cap access(d_n)\}$$

Simply, above relationship means that each and every individual i representing role r, such that $r = g$ and $g \in access(agenda)$, is also an element of the

intersection of $access(d)$ of all private documents discussed in meeting. A situation in which above relationship is violated can be explained by using following relationship.

$$\{access(d_1) \cap access(d_2) \cap ... \cap access(d_{n-1}) \cap access(d_n)\} \subset access(agenda)$$

In simple terms, above relationship means that there exists at least one i representing r, such that $r = g$ and $g \in access(agenda)$, where g is not an element in the intersection of $access(d)$ of all private documents discussed in meeting.

However, for discussing public documents in meeting, it is not required to perform any participant validation. In other words, any individual i can discuss public documents, in any private or public meeting.

9 Privacy-preserved meeting

Based on above descriptions and definitions, we define privacy-preserved meeting as below;

A privacy-preserved meeting is a meeting in which, individual i representing role r has no access to the meeting, when $r = g$ **and** $g \notin access(agenda)$**, where access(agenda) satisfies,**

$$access(agenda) \subseteq \{access(d_1) \cap access(d_2) \cap ... \cap access(d_{n-1}) \cap access(d_n)\}$$

for all private documents $d_1...d_n$**, discussed in the meeting.**

10 Meeting quorum

In a **privacy-preserved meeting** of our research context, we define **meeting quorum** as the minimum number of individuals (i) representing participant groups (g), required to attend a meeting, such that $g \in access(agenda)$.

In privacy preserved meeting context, if a specific **meeting quorum** rule is not defined in the agenda, other than $access(agenda)$ set, we assume that every i such that,

- $i \in g$, and
- $g \in access(agenda)$,

is required for the meeting. But, it is not applicable for $g =$ **public**. In such a situation, when a numeric meeting quorum rule is not defined specifically, $|meeting\ quorum| = |access(agenda)|$, where $access(agenda)$ is defined in form of singleton subsets of $\mathcal{G}$.

In addition, it is possible that $|meeting\ quorum| < |access(agenda)|$, if a numeric **meeting quorum** rule is defined in meeting agenda. Therefore in overall, $|meeting\ quorum| \leq |access(agenda)|$, when $access(agenda)$ is defined in form of singleton subsets of $\mathcal{G}$.

Since at least 2 individuals (i) are required for any meeting, $2 \leq |meeting\ quorum|$.

Accordingly, $2 \leq |meeting\ quorum| \leq |access(agenda)|$.

When $access(agenda)$ is defined in form of singleton subsets of $\mathcal{G}$, as we have already depicted earlier, $|access(agenda)| \leq |\mathcal{I}|$. By merging this inequality with above expression, we obtain following expression theoretically.

$2 \leq |meeting\ quorum| \leq |access(agenda)| \leq |\mathcal{I}|$

11 Problem analysis

Our problem focused in organizing privacy-preserved meetings has few distinct steps, that can be clearly identified within it. Based on decision making points identified within the problem, this problem was mapped into a boolean circuit, following a union operation. Here, union operation can be introduced as a set related pre-processing operation, applied on the $access(d)$ sets of all documents of the meeting, inlcuding the *agenda*. Before applying this union operation, it is needed to make sure that, all groups except the *public* group in those $access(d)$ sets are converted to singleton groups. Distinct sections of the problems are analyzed below, considering the circuit diagrams produced for them.

11.1 Participant validation based on documents

After calculating the union of $access(d)$ sets of all documents to be discussed in the meeting, as $access(doc\_1) \cap ... \cap access(doc\_n)\} \cap access(agenda)$, *public* group is subtracted, since our initial requirement is to identify all the individuals having access to at least one document. In addition, we identify all documents associated with meeting as $doc\_1, ..., doc\_n, agenda$, since we need to check whether each individual in the union of individuals identified, has access to all the documents, for discussing them in a meeting. In example depicted in the diagram, consider that there are only 3 individuals as i_1, i_2, i_3 and only 4 documents as $doc\_1, doc\_2, doc\_3, agenda$.

Algorithm for participant validation section is explained below.

Algorithm 1 Participant validation based on documents

Input: Access control lists (access(d)) of documents doc_1, ... , doc_n, $agenda$
Output: Eligibility of each individual for meeting by document analysis

union of $access(d) = access(doc\_1) \cup \cdots \cup access(doc\_n) \cup access(agenda)$
union of individuals = union of $access(d) - public$
set of documents = $doc\_1, \ldots, doc\_n, agenda$
for each i_n in (union of individuals) **do**
    validity of i_n by doc analysis = true
    for each doc_x in (set of documents) **do**
        if $doc\_x \neq agenda$ **then**
            validity of i_n for $doc\_x = (i\_n \in access(doc\_x))$ OR $(public \in access(doc\_x))$
        else
            validity of i_n for $doc\_x = i\_n \in access(agenda)$
        end if
        validity of i_n by doc analysis = validity of i_n by doc analysis AND validity of i_n for doc_n
    end for
    Return validity of i_n by doc analysis
end for

Above algorithm explains the process depicted by logical circuit diagram below. In the algorithm,

$$|union\ of\ access(d)| = |access(doc\_1)| + ... + |access(doc\_n)| + |access(agenda)|$$

relationship means that time complexity of the $union\ of\ access(d)$ opearion is, $O(|access(doc\_1)| + ... + |access(doc\_n)| + |access(agenda)|)$. It is a linearly increasing time complexity. In addition, each iteration in outer loop outputs whether an individual of the union of individuals is valid by $access(d)$ analysis of all documents. Inner loop checks whether particular individual has access to all documents including $agenda$. Operations within the loops are considered as operations of constant time complexity, $O(1)$. Accordingly, since there is a nested loop in above algorithm, time complexity of this section of problem can be $O(n^2)$, at maximum. Because number of individuals can be any positive integer, and number of documents including $agenda$ also can be any positive integer.

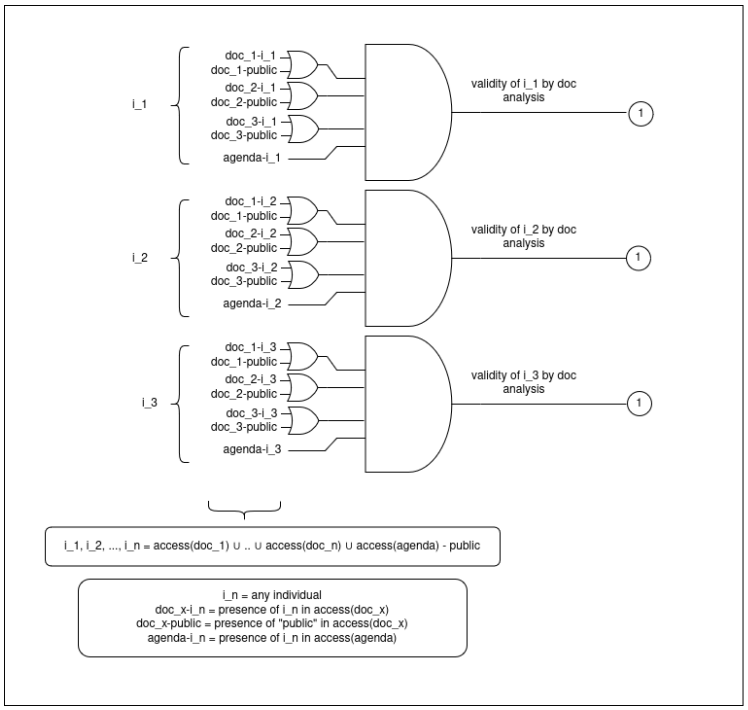

Figure 3: Participant validation based on $access(d)$ of documents

11.2 Meeting quorum satisfiability of time slots

11.2.1 Eligibility of each individual, in each time slot, for meeting

After checking the validity of each participant by document analysis, it is needed to check their locations in different time slots. Based on the location, eligibility of each individual to discuss the set of documents changes. Because, as we have described earlier, *private* documents should be discussed at *private* locations only, while *public* documents can be discussed at any location. If there is at least one *private* document in the set of documents discussed in meeting, then each individual i_n validated by document analysis should attend this meeting from a *private* location. Following circuit diagram consists of the location analysis of each individual, in every time slot. Here, these time slots mean the time slots belonging to the union of available time slots of all individuals, present in the union of individuals, mentioned in previous algorithm.

For deciding the eligibility of each individual, to discuss set of documents, in different time slots, we need inputs included in the following table. In it, each individual of the union of individuals, and his/her available time slots with locations should be present. Please note that following table consists of some sample data only, as individuals, time slots and locations. It is supposed to contain the

real data, regarding the respective scenario. For simplicity, we consider only 3 locations as onsite, remote_private and remote_public, since every location can be categorized into one of those 3 categories, by a meeting organizing entity.

Individual (i_x)	Time slot(slot$y$)	Location
i_1	slot1	onsite
i_1	slot2	remote_private
i_1	slot3	remote_public
...	...	...
i_n	slot$m$	location of i_n in slot$m$

Table 2: Available time slots and locations of individuals, in union of individuals

Algorithm for deciding the eligibility of each individual, to discuss the set of documents of meeting, in different time slots, is described below.

Algorithm 2 Deciding eligibility of each individual for meeting, in each time slot

Input 1: Validity of each individual i_1, ... , i_n by doc analysis
Note: Individuals i_1, ... , i_n mean union of individuals of previous algorithm, and **input 1** is obtained from output of previous algorithm
Input 2: Availability of each individual in each time slot (at any location)
Input 3: Location of each individual in each time slot (whether remote_public or no)
Note: Input 2 and **input 3** are obtained from the table containing available time slots and locations of individuals.
Input 4: Presence of *public* group in $access(d)$ of each document, in set of documents, as $(doc\_1 - public), ..., (doc\_n - public), (agenda - public)$
Output: Eligibility of each individual, to discuss the set of documents, in each time slot

union of time slots = $availability(i\_1) \cup \cdots \cup availability(i\_n)$
for each slot$x$ in (union of time slots) **do**
  for each i_y in (union of individuals) **do**
    publicity of meeting = (doc_1 − *public*) AND ... AND (doc_n − *public*) AND ($agenda - public$)
    slot$x - i_y$ = availability of i_y in slot$x$ at any location
    slot$x - i_y$_remote_public = whether i_y is at remote_public location in slot$x$
    availability of slot$x - i_y$ for public meeting = (publicity of meeting) AND (slot$x - i_y$)
    availability of slot$x - i_y$ for private meeting = (slot$x - i_y$) AND NOT(slot$x - i_y$_remote_public)
    availability of slot$x - i_y$ for meeting by time slot analysis = (availability of slot$x - i_y$ for public meeting) OR (availability of slot$x - i_y$ for private meeting)
    slot$x - i_y$ eligibility for meeting = (Validity of i_y by doc analysis) AND (availability of slot$x - i_y$ for meeting by time slot analysis)
  end for
  Return slot$x - i_y$ eligibility for meeting
end for

In above algorithm, time complexity of *union of time slots* operation is, $O(|availability(i\_1)| + \cdots + |availability(i\_n)|)$, at maximum. It is a linear time complexity. When considering the loops, outer loop iterates through all time slots, in the union of time slots, calculated by considering available time slots of all individuals, in union of individuals. Meanwhile, inner loop iterates through each individual in the union of individuals. In addition, number of documents discussed in the meeting can be any positive integer. Number of documents is considered when deciding the meeting publicity, within the inner loop. Further, validity of the individual by document analysis, obtained from output of previ-

ous algorithm, is utilized in in inner loop. Number of documents is concerned for that as well. But, it is in parallel level, with publicity calculation of the meeting. Therefore, we can consider that number of documents has linear time complexity as $O(n)$, within inner loop. Accordingly, it is observed that, time complexity of above algorithm can be $O(n^3)$, at maximum, due to outer loop, inner loop, and number of documents within inner loop.

Circuit diagram corresponding to above algorithm is depicted below. The following boolean circuit is supposed to be repeated for each individual, in each time slot, as explained in algorithm.

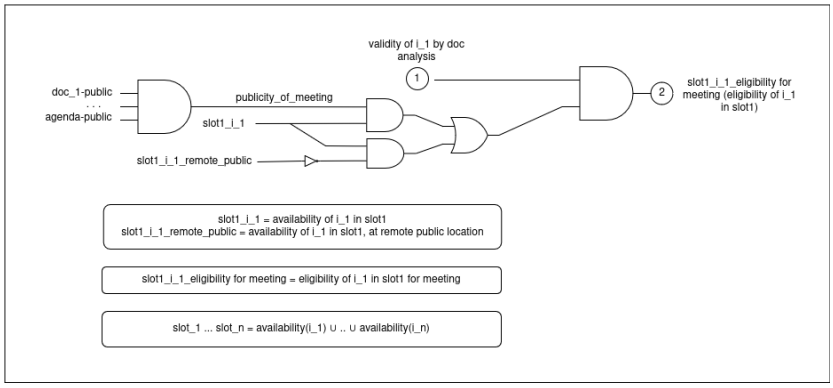

Figure 4: Eligibility of each individual for meeting, in each time slot

11.2.2 Meeting quorum satisfiability

Eligibility of each individual to discuss the set of documents, in each time slot, is considered for deciding the quorum satisfiability of each time slot, in the union of time slots. Then it is checked whether there is at least one quorum satisfying time slot. Because, if there is no at least one quorum satisfying time slot, then it is not possible to conduct the meeting.

Algorithm for checking the quorum satisfiability of time slots is expalined below.

Algorithm 3 Identifying meeting quorum satisfiability of time slots

Input 1: Eligibility of each individual for meeting, in each time slot
Note: Input 1 is obtained from output of previous algorithm
Input 2: Numerical meeting quorum value
Input 3: Availability of each individual in each time slot (at any location)
Output: Existence of quorum satisfying time slot/s, for the meeting

union of time slots = $availability(i\_1) \cup \cdots \cup availability(i\_n)$
availability of a meeting quorum satisfying time slot = false
for each slot$x$ in (union of time slots) **do**
    meeting quorum satisfiability of slot$x$ = false
    combinations of individuals in slot$x$ = combinations of individuals, with magnitude of meeting quorum size
    for each *combination* in (combinations of individuals in slot$x$) **do**
        meeting quorum satisfiability of *combination* = AND operation (for all individuals in the *combination*)
        meeting quorum satisfiability of slot$x$ = (meeting quorum satisfiability of slot$x$) OR (meeting quorum satisfiability of *combination*)
    end for
    Return availability of a meeting quorum satisfying time slot = (availability of a meeting quorum satisfying time slot) OR (meeting quorum satisfiability of slot$x$)
end for

Similar to previous algorithm, in above algorithm also, time complexity of *union of time slots* operation is, $O(|availability(i\_1)|+\cdots+|availability(i\_n)|)$, at maximum. It is observed that, in loop structure, outer loop iterates through all time slots in union of time slots. That number of time slots can be any positive integer. Inner loop contains a *combination* operation, which is intended to create combinations of magnitide of meeting quorum size, out of individuals of the union of individuals. Mathematically, number of such possible combinations can be calculated as,

$$n\mathrm{C}r = \binom{n}{r} = \frac{n!}{r!(n-r)!}$$

when n = number of individuals in union of individuals, and r = meeting quorum size.

Regarding *combination* operation, highest number of combinations is obtained when $r = \frac{n}{2}$ or $r \approx \frac{n}{2}$. Number of possible combinations reduces, when r is considerably small ($r \approx 1$), or when r is approximately equal to n ($r \approx n$). Due to this nature of *combination* operation, maximum possible time complexity for creating $n\mathrm{C}r$ number of combinations is $O(n\mathrm{C}(n/2))$.

In the inner loop of above algorithm, since AND operation is applied for all individuals, in each *combination* created, maximum time complexity associated

with inner loop is $O(n \text{ x } n\text{C}(n/2))$. Because, number of individuals in each combination created, or the meeting quorum size, can be any positive integer. In addition, since inner loop is repeated by outer loop, where number of time slots also can be any positive integer, maximum possible time complexity of above algorithm is,

$$O(n \text{ x } n \text{ x } n\text{C}(n/2)) = O(n^2 \text{ x } n\text{C}(n/2))$$

Time complexity of $O(n^2 \text{ x } n\text{C}(n/2))$ has polynomial component of $O(n^2)$. Further, when considering $O(n\text{C}(n/2))$ component, it is observed that, for large n values,

$$O(n\text{C}(n/2)) \approx O(2^n)$$

Time complexity of $O(2^n)$ is considered as an exponential time complexity. Since exponential time complexity is non-polynomial (NP), it is eligible to use a heuristic or an existing library, when implementing above algorithm for execution with inputs. Because, there is no specific standard methodology defined, for creating combinations, as included in above algorithm.

Figure below depicts the corresponding boolean circuit for above algorithm.

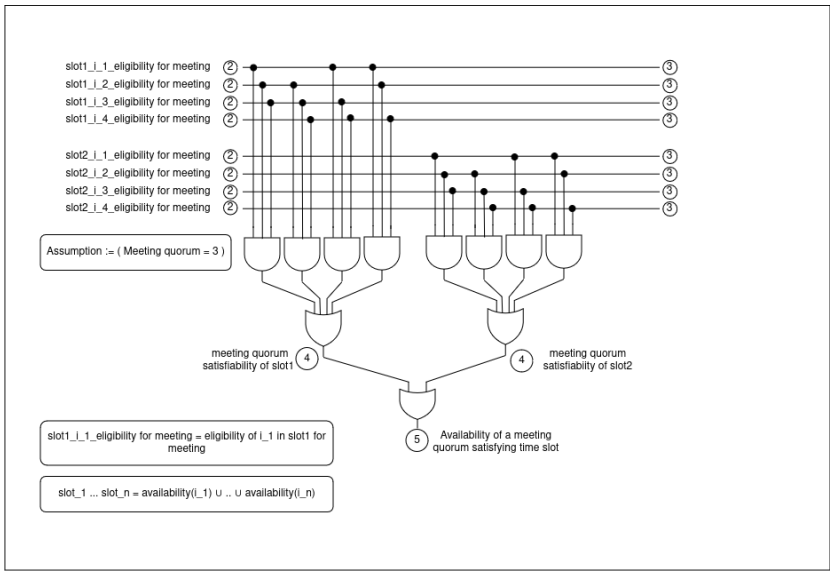

Figure 5: Meeting quorum satisfiability of time slots

11.3 Selection of earliest, meeting quorum satisfying time slot

Selecting the earliest, meeting quorum satisfying time slot is not required to find a solution for our research problem. Because, after identifying the quorum satisfying time slots by above algorithm, our next objective associated with research problem is to, analyze the privacy-preserving meeting mode for each meeting quorum satisfying time slot. Therefore, we introduce this step of selecting the earliest, meeting quorum satisfying time slot, as an additional step provided by us. Since privacy-preserved meeting organization is a real world problem, when implementing a usable system for that purpose, this additional step can be helpful to identify the ealiest, eligible time slot for conducting the privacy-preserved meeting, rather than suggesting numerous eligible time slots.

Algorithm for selecting the earliest, meeting quorum satisfying time slot is explained below.

Algorithm 4 Selection of earliest, meeting quorum satisfying time slot

Input: Meeting quorum satisfiability of time slots, in union of time slots
Note: Input is produced by processing previous algorithm
Output: Earliest, meeting quorum satisfying time slot

union of time slots with quorum satisfiability = quorum satisfiability of each time slot, in $(availability(i\_1) \cup \cdots \cup availability(i\_n))$, in the order of precedence
earliest eligible time slot = initialization of an empty list type structure, to store whether each slot is earliest eligible slot or no
for each slot$x$ and $meeting\_quorum\_satisfiability$ in (union of time slots with quorum satisfiability) **do**
  if slot$x$ is earliest slot in (union of time slots with quorum satisfiability) **then**
    store slot$x$ and its $meeting\_quorum\_satisfiability$, in (earliest eligible time slot) list
  else
    previous condition = list to store result of, NOT operation (for each slot currently stored in (earliest eligible time slot) list)
    previous condition integrated = AND operation (for all slots in (previous condition) list)
    whether slot$x$ is earliest meeting quorum satisfying time slot = (($meeting\_quorum\_satisfiability$ of slot$x$ = true) AND ((previous condition integrated) = true))
    store slot$x$ and (whether slot$x$ is earliest meeting quorum satisfying time slot), in (earliest eligible time slot) list
  end if
end for
Return (earliest eligible time slot) list
Note: At **Return** point, (earliest eligible time slot) list contains *true* for earliest, meeting quorum satisfying time slot, and $false$ for all other time slots

In above algorithm, there is a reason to apply a different process on the earliest time slot in union of time slots, apart from all other time slots in the union. If earliest time slot in union of time slots satisfies the meeting quorum, then without considering any pre-condition, it becomes the earliest meeting quorum satisfying time slot. It can be analyzed by extracting it as a separate condition, like below.

If earliest slot in union of time slots satisfies the meeting quorum, **Then**
  earliest slot is the earliest, meeting quorum satisfying time slot
Else
  earliest slot is **not** the earliest, meeting quorum satisfying time slot
EndIf

Above condition block is the meaning implied by "If" block, in the condition of the algorithm. But, if any time slot other than the earliest time slot of union, satisfies the meetng quorum, then it is needed to check an additional condition as well, to identify whether it is the **earliest** meeting quorum satisfying time slot or no. Accordingly, following condition block elaborates the meaning implied by "Else" block of the condition, in above algorithm.

If any slot$x$ which is **not** the earliest slot in union of time slots, satisfies the meeting quorum, **Then**
    If any earlier slot than slot$x$ does **not** satisfy the meeting quorum, **Then**
        slot$x$ is the earliest, meeting quorum satisfying time slot
    EndIf
Else
    slot$x$ is **not** the earliest, meeting quorum satisfying time slot
EndIf

It is observed that, there is a nested "If" condition in "If" block, in above condition block. Requirement to check this additional condition, is the reason for treating earliest time slot in union of time slots in a different manner, than other time slots, when identifying the earliest meeting quorum satisfying time slot.

Above algorithm utilizes the meeting quorum satisfiability of each time slot, obtained by processing previous algorithm. Therefore, When we calculate the time complexity of above algorithm, we consider that meeting quorum satisfiability information is stored in an eligible data structure after processing previous algorithm. Otherwise, if we consider these both algorithms together, time complexity of above algorithm and previous algorithm collectively becomes, $O(2^n)$ at maximum, as explained in time complexity description of previous algorithm.

But, when meeting quorum satisfiability information obtained from previous algorithm is stored in an eligible data structure and provided to above algorithm, time complexity of above algorithm is dominated by "for" loop, and its content. Number of iteration in this loop can be any positive integer. There is no inner loop, in the "for" loop. But, in "Else" block of each iteration, "NOT" operation is applied on each time slot, until the slot which is currently being iterated by "for" loop. Therefore, it is observed that, time complexity of above algorithm can be $O(n^2)$, at maximum. Even though there is an "AND" operation applied on all time slots, in the "Else" block, it can be neglected, when calculating time complexity. Because, it is in the parallel level to "NOT" operation, which is already considered. All other operations within above algorithm are considered as operations having constant time complexity.

Circuit diagram corresponding to above algorithm is depicted below.

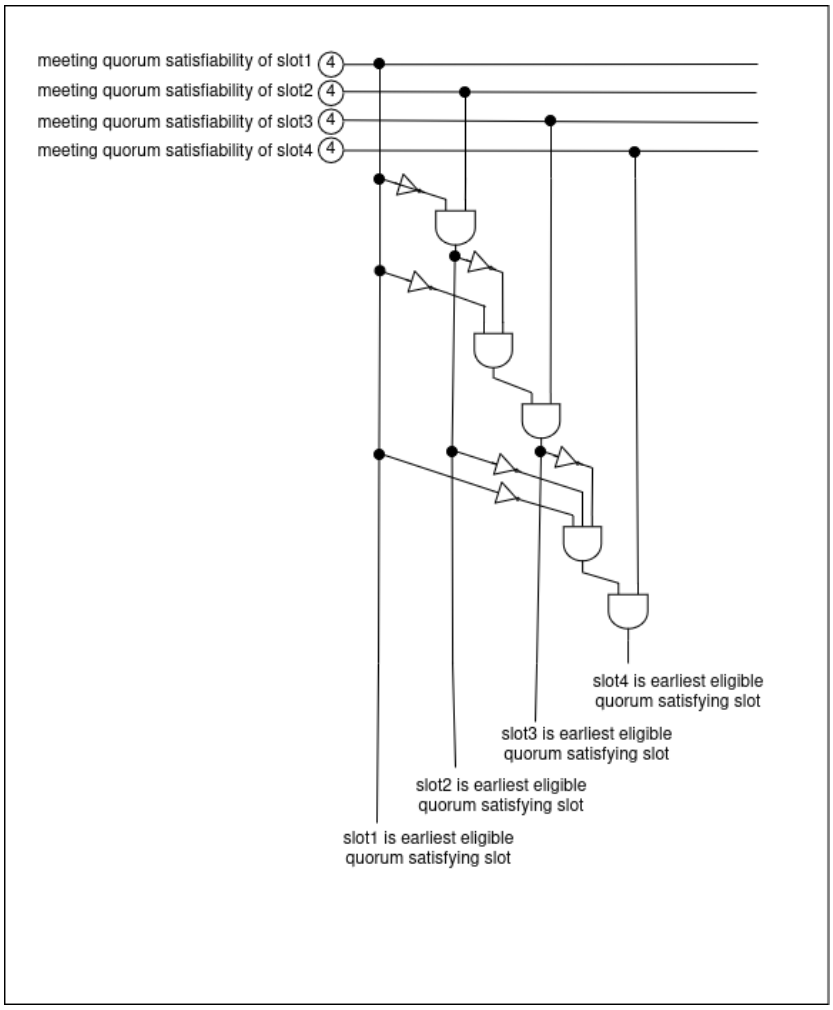

Figure 6: Selection of earliest, meeting quorum satisfying time slot