Privacy-Preserved Meeting Organization

Group 06

February 5, 2025

1 Introduction

Effective and privacy-preserved communication has received a significant concern in the world. As remote work, hybrid teams along with global collaboration has risen, the meetings are playing an important part in organizations. The shift to virtual and hybrid environments has created new and unique challenges when ensuring that meetings are privacy-preserved and inclusive. Meeting organizers face difficulties in allowing accessibility to participants while maintaining privacy of the meeting especially when sensitive meetings take place with sensitive documents and diverse participant roles. This brings our attention to the need for a better framework that can enhance the meeting scheduling process without compromising meeting privacy and efficiency.

When meetings are organized, it must be well planned ahead of what is being discussed and what needs to be achieved at the end of the meetings. If the right participants are not included or if inappropriate participants are included in the meeting, that can lead to privacy concerns and even to potential breaches of confidential information.

Before the COVID-19 pandemic, meeting security and privacy concerns were relatively minimal, as most discussions took place in controlled, onsite environments. But the pandemic reshaped how organizations conduct meetings increasing the reliance on online and hybrid formats. Since 2020, the adoption of virtual meeting platforms has raised to accommodate remote work and collaboration (Jo et al., 2021). While this shift has improved accessibility and flexibility, it has also introduced significant privacy challenges.

One of the major problems is in allowing only authorized participants to attend the meeting. Human cognitive biases and organizational pressures can lead to oversights in meeting privacy, which can result in financial losses, reputational damage, and unauthorized disclosure of sensitive information. Selecting the right participants and ensuring that meeting-related documents remain accessible only to authorized individuals are critical aspects of maintaining privacy and security. Existing tools lack the ability to cross-validate document permissions with participant roles and their joining locations. This leads to potential meeting privacy

breaches. For instance, if participants are joining from insecure public locations, sensitive discussions may unintentionally become exposed. The inability to determine whether a meeting should be onsite, online, or hybrid based on such parameters increase organizational inefficiencies and privacy risks.

Research on meeting scheduling and efficiency has been conducted from a long time but no attention has been given to privacy-preserving meeting organization. Existing scheduling tools focus on preventing participant overlap or optimizing time slots but fail to consider how document permissions and participant roles influence meeting security.

Our research aims to explore the relationship between meeting privacy factors such as document access, participant roles, and meeting mode and develop a structured framework to ensure meeting privacy. The significance of this research is in its dual contribution. At first, the theoretical understanding of meeting scheduling complexity is studied in-depth, and next, a practical implementation using widely adopted tools is demonstrated. By combining algorithmic analysis with a working prototype, this study provides insights into the computational nature of meeting organization and offers a scalable, privacy-preserving approach to organize meetings efficiently. This thesis contributes to both academic and practical understanding of organizing privacy-preserved meeting organization.

2 Problem Definition

When multiple topics are to be discussed, it is challenging to determine the right participants for the meeting, and meeting organizers can include the wrong participants in the meeting. This can lead to two primary meeting privacy risks which we will be addressing when developing our framework:

Unauthorized Access to Confidential Information: When the meeting organizer has invited unauthorized participants to the meeting, they may gain access to restricted documents and be exposed to sensitive discussions that they are not authorized to view or hear. Here we can see the need for a systematic approach to invite eligible participants or participants with access rights to the meeting.

Privacy-Violating Meeting Modes: We often see how meetings are organized by meeting organizers based on their convenience or on the convenience of people in the upper hierarchy. This can lead sensitive discussions to be exposed to unauthorized individuals. The meeting mode decision-making process should be systematic based on the participants' locations and the sensitivity of the meeting agenda.

Based on the above privacy concerns, a need for a more structured approach to meeting organization can be identified as a crucial need.

In this research we assume that the meeting participant selection is primarily determined by ACLs of the documents to be discussed. Once the meeting organizer creates the meeting agenda with the participants, it is important to validate the participants based on the access controls to the documents that are being taken to the particular meeting. If the participants do not meet the required access permissions, the meeting cannot be scheduled as planned.

Once participant selection is validated, the appropriate meeting mode is determined next. This decision depends on the locations of the eligible participants and the sensitivity of the meeting. Finally, the system selects the earliest possible time slot that allows all participants to join while maintaining meeting privacy. The research problem is therefore to determine the algorithms that schedule meetings while validating the participants and taking privacy-aware mode into consideration.

3 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.

4 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.

5 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**.

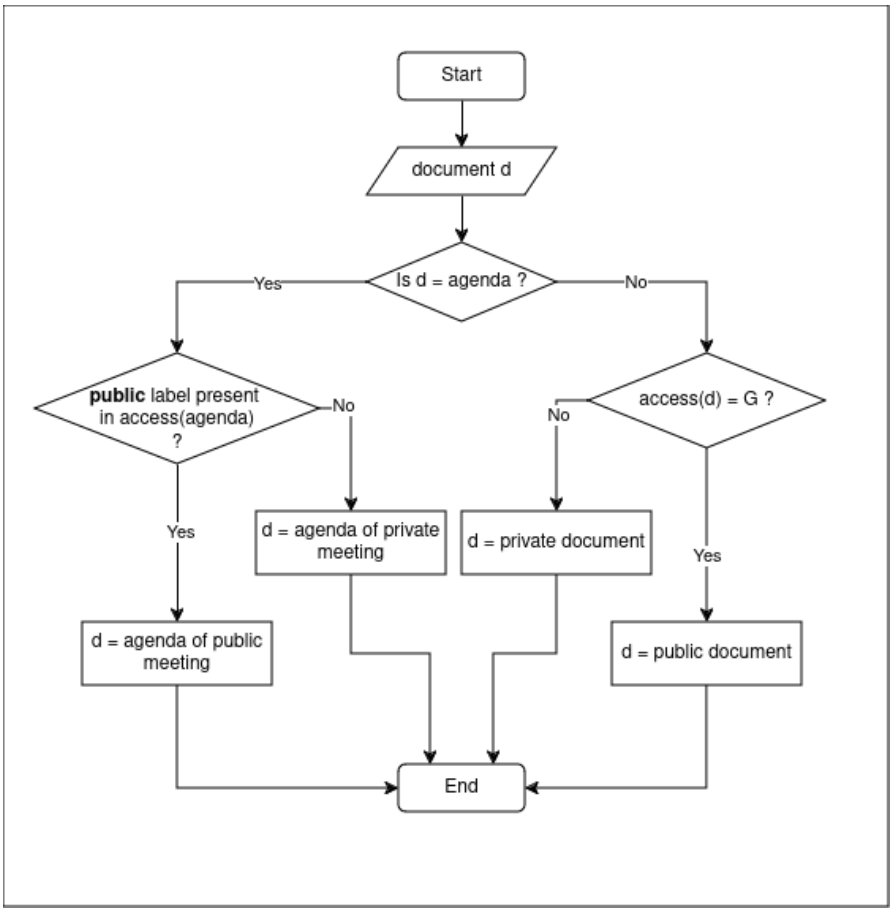

Figure 1: Process to identify whether a document is private or public

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.

6 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 individuals 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 **synchronous** 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-tuple,

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

such that,

$$D \subseteq \mathcal{D}$$
$$L \subseteq \mathcal{L}$$
$$I \subseteq \mathcal{I}$$
$$t \in \mathcal{T}$$

7 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.

8 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.

8.1 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.

9 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.

10 Privacy of documents

10.1 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 constraints to protect 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 individual 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.

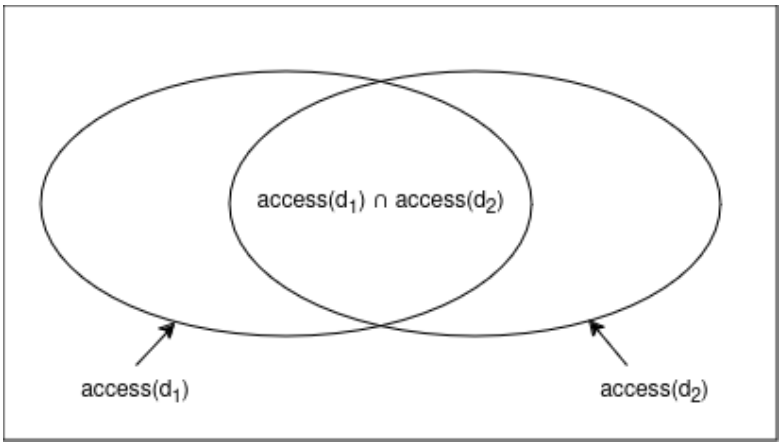

Figure 2: Intersection of access control lists of 2 private documents

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.

10.2 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.

11 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.**

12 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.

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

13 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, including 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.

13.1 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 individual 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 $union\ of\ access(d)$ operation 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.

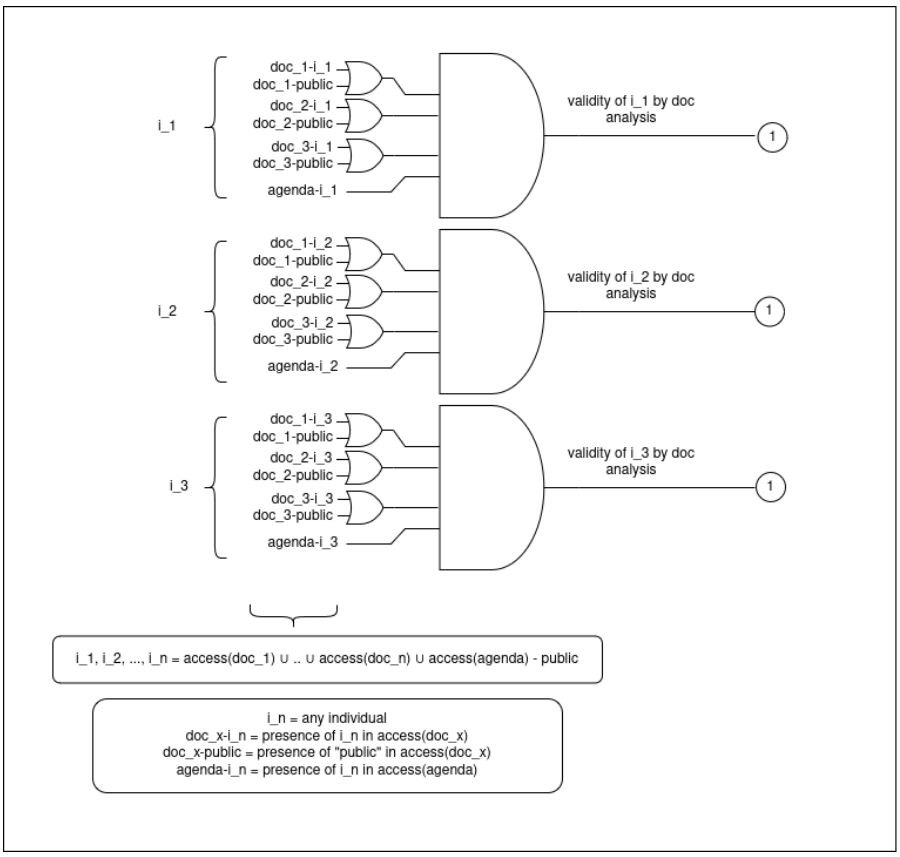

Figure 3: Participant validation based on $access(d)$ of documents

13.2 Meeting quorum satisfiability of time slots

13.2.1 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 individual 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 previous 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.

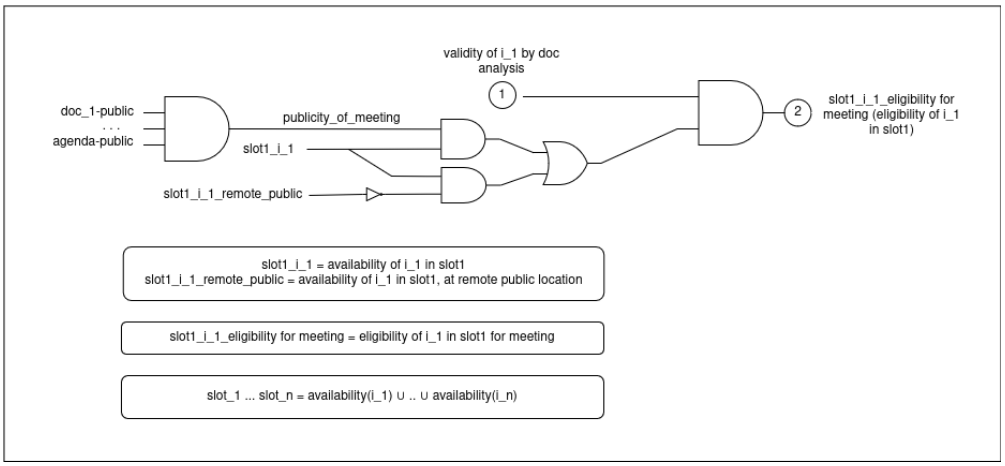

Figure 4: Eligibility of each individual for meeting, in each time slot

13.2.2 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 explained 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 magnitude 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 a non-polynomial time complexity, because n is not a constant value (Pfleeger 1996). Since non-polynomial time complexity of $O(2^n)$ increases exponentially with increase of n, 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.

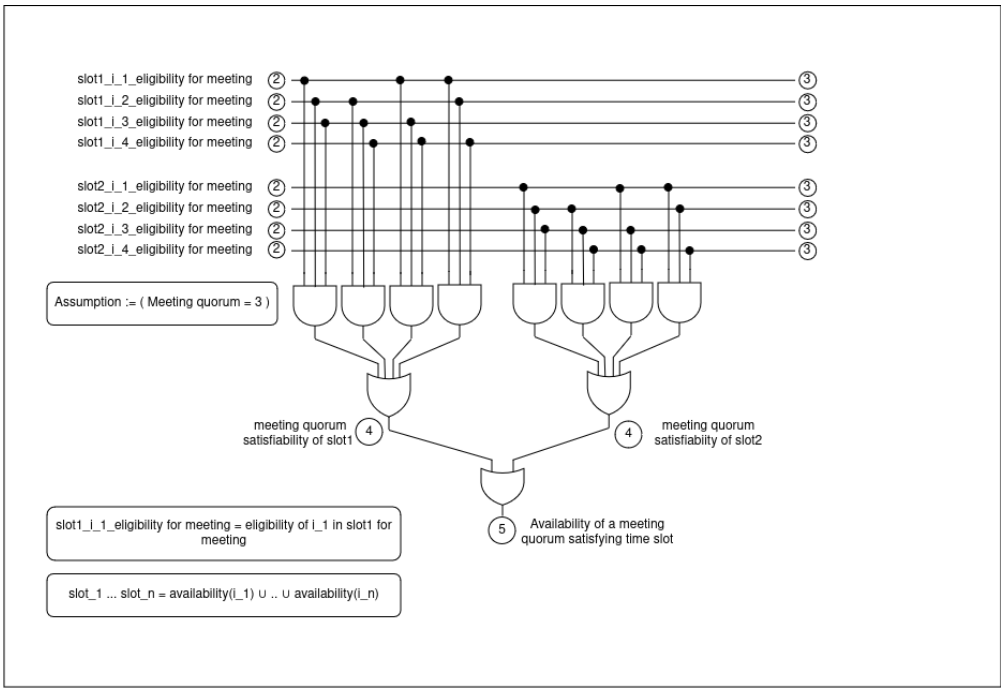

Figure 5: Meeting quorum satisfiability of time slots

13.3 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 earliest, 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
End If

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 meeting 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
    End If
Else
    slot$x$ is **not** the earliest, meeting quorum satisfying time slot
End If

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.

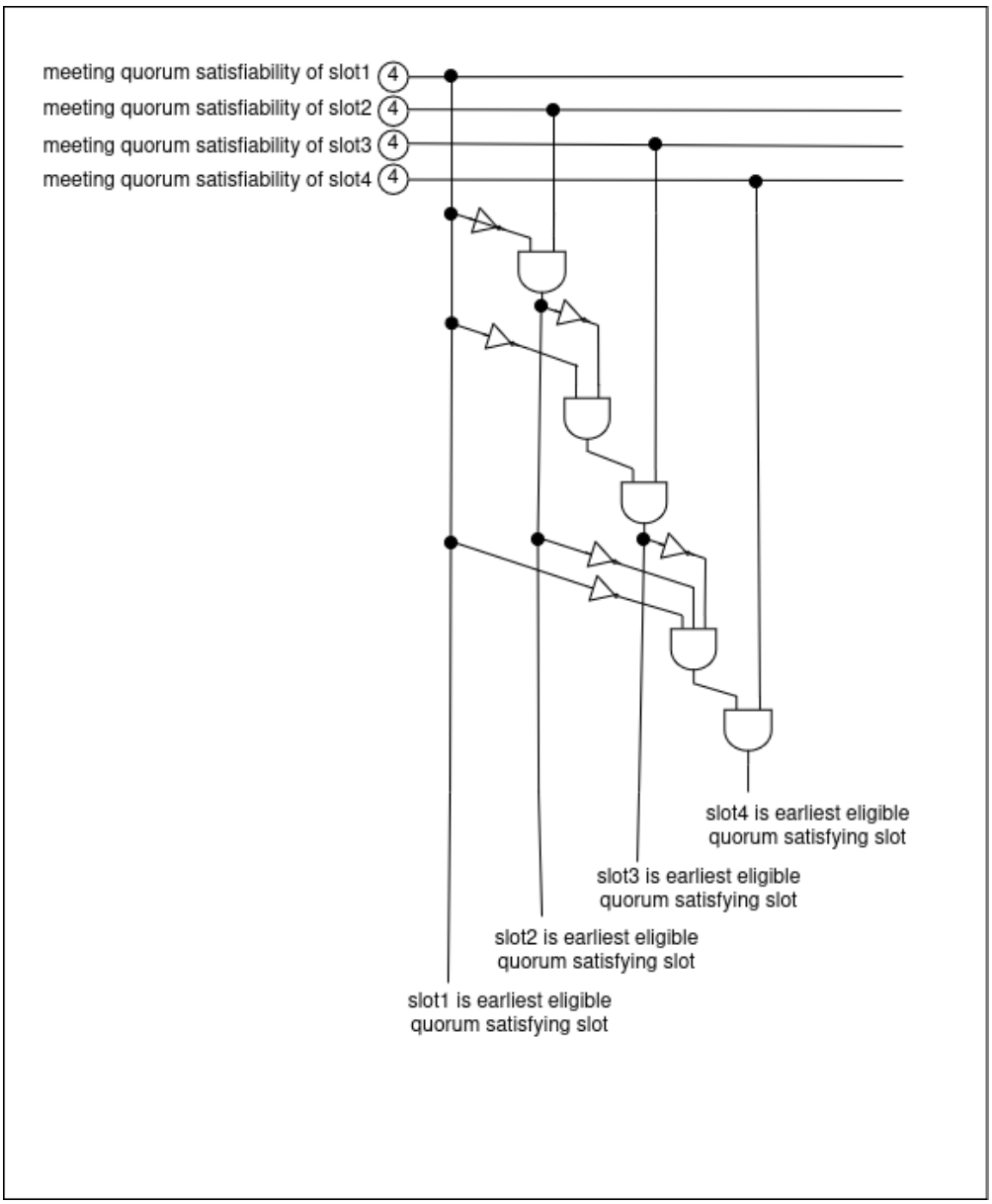

Figure 6: Selection of earliest, meeting quorum satisfying time slot

13.4 Meeting mode selection

After identifying the quorum satisfying time slots, meeting mode of each time slot should be decided, based on locations of eligible participants. In real world scenario, participants can be present at numerous locations. However in our research, as it is already mentioned earlier, only three location types are considered (onsite, remote_private and remote_public locations). It is observed that three meeting modes are possible, based on those three locations, as onsite mode, hybrid mode and online mode. Logic for deciding the meeting mode is depicted by the flow chart below.

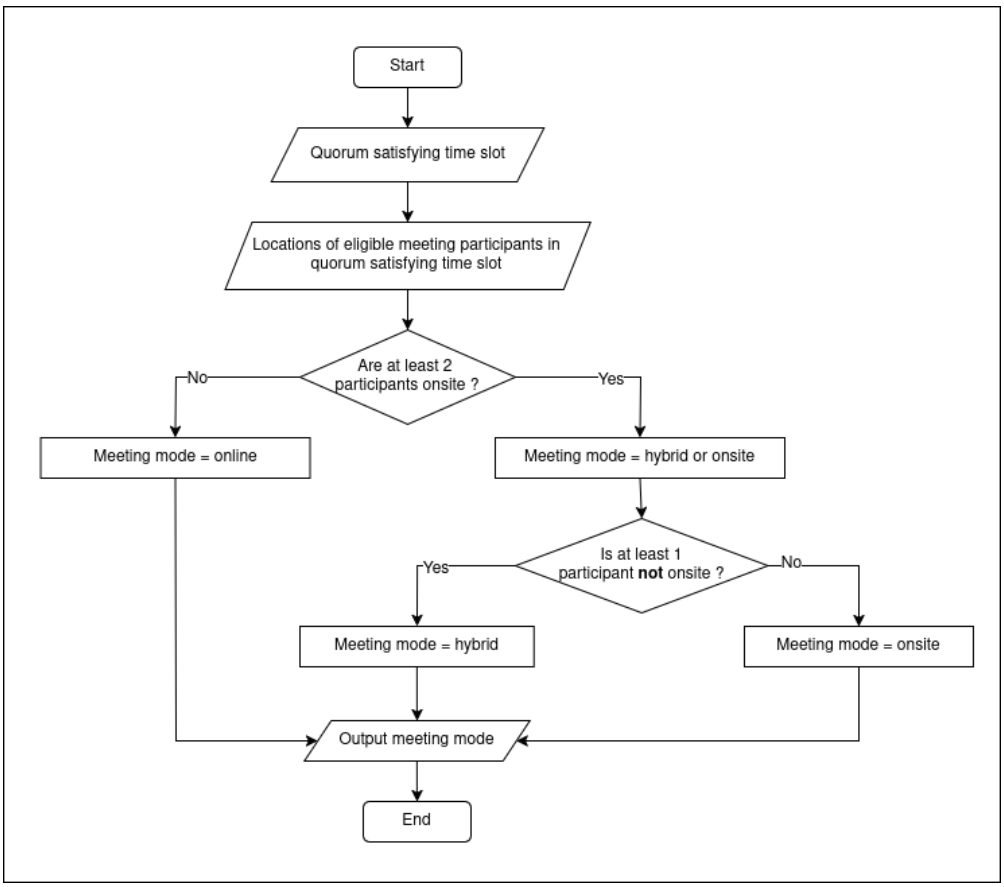

Figure 7: Logic for selecting the meeting mode

According to the logic, initially we check whether at least two eligible participants are at onsite locations, during the quorum satisfying time slot. If only one participant is at onsite location (i.e. onsite office location of meeting organization), or no one is at onsite location, it means that obviously meeting will be in online mode. But, if at least two participants are available at onsite location, it is required to check whether at least one participant is not at the onsite location. If at least one participant is at a remote location in the considered time slot, meeting will obviously be in hybrid mode. Unless, if at least two participants are onsite and, at least one participant is **not** at a remote location (remote_private or remote_public), it means that all available participants are at onsite location. It implies that meeting is in onsite mode.

The algorithm for deciding the meeting mode of quorum satisfying time slot is explained below.

Algorithm 5 Meeting mode selection

Input 1: Eligibility of each participant for meeting, in each time slot
Note: Input 1 is produced by processing algorithm 1
Input 2: Meeting quorum satisfiability of each time slot, in union of time slots
Note: Input 2 is produced by processing algorithm 3
Input 3: Whether each meeting participant is at onsite location, in each time slot
Output: Whether meeting mode is onsite or hybrid or online

for each slot_x in (union of time slots) **do**
slot_x onsite = initialize an empty list
slot_x online = initialize an empty list
for each participant_y with *eligibility for meeting* status in slot_x **do**
participant_y_slot_x_onsite = whether participant_y is onsite in slot_x
append to (slot_x onsite) list: (*eligibility for meeting* for participant_y) AND (participant_y_slot_x_onsite)
append to (slot_x online) list: (*eligibility for meeting* for participant_y) AND NOT(participant_y_slot_x_onsite)
end for
slot_x is onsite or hybrid = false
onsite combinations of slot_x = create all possible combinations of size 2, with items in (slot_x onsite) list
for each combination_z in (onsite combinations of slot_x) **do**
combination_z is onsite = apply AND operation between 2 items in combination_z
slot_x is onsite or hybrid = (slot_x is onsite or hybrid) OR (combination_z is onsite)
end for
eligibility of slot_x for online mode = (NOT(slot_x is onsite or hybrid)) AND (meeting quorum satisfiability of slot_x)
slot_x is hybrid = false
for each participant p in (slot_x online) list **do**
slot_x is hybrid = (slot_x is hybrid) OR (whether participant p is online)
end for
eligibility of slot_x for hybrid mode = (slot_x is hybrid) AND {(meeting quorum satisfiability of slot_x) AND (slot_x is onsite or hybrid)}
eligibility of slot_x for onsite mode = (NOT(slot_x is hybrid)) AND {(meeting quorum satisfiability of slot_x) AND (slot_x is onsite or hybrid)}
Return eligibility of slot_x for online mode, eligibility of slot_x for hybrid mode, eligibility of slot_x for onsite mode
Note: At **Return** point, depending on whether eligible meeting mode is online or hybrid or onsite, respective variable will be **true**, and other two variables will be **false**.
end for

When considering the algorithm above, it is observed that it obtains multiple outputs from previous algorithms. In other words, it is impossible to provide inputs required by this algorithm, without processing previous algorithms. However, since it is not mandatory to execute this algorithm in parallel with previous algorithms, it is possible to calculate the the time complexity of this algorithm, as below.

In above algorithm, it is observed that there is a nested "for" loop structure. There are some separate"for" loops, inside the outer loop. Among those "for" loops, the loop iterating through all possible combinations of two individuals has $n\mathrm{C}r$ number of iterations, where n = number of individuals in union of individuals, and r = 2. Therefore, maximum possible number of iterations in that particular inner loop can be mathematically calculated as below.

$$n\mathrm{C}2 = \binom{n}{2} = \frac{n!}{2!(n-2)!}$$

$$\frac{n!}{2!(n-2)!} = \frac{n!}{2 \text{ x } (n-2)!}$$

$$\frac{n!}{2 \text{ x } (n-2)!} = \frac{(n-2)! \text{ x } (n-1) \text{ x } n}{2 \text{ x } (n-2)!}$$

$$\frac{(n-2)! \text{ x } (n-1) \text{ x } n}{2 \text{ x } (n-2)!} = \frac{(n-1) \text{ x } n}{2}$$

$$\therefore n\mathrm{C}2 = \frac{(n-1) \text{ x } n}{2} = \frac{n^2}{2} - \frac{n}{2}$$

Maximum possible time complexity of "for" loop, which iterates through all possible combinations of two individuals is $O(n\mathrm{C}r)$. Since value of r = 2, and depending on above calculation, time complexity of this "for" loop can be depicted as below.

$$O(n\mathrm{C}2) = O(\frac{n^2}{2} - \frac{n}{2})$$

In addition, when considering the outer "for" loop together with the inner "for" loop, that iterates through all possible combinations of two individuals, total time complexity can be calculated as below. Because, outer loop can iterate through n number of iterations, where n can be any positive integer.

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

$$O(n \text{ x } (n\mathrm{C}2)) = O(n \text{ x } (\frac{n^2}{2} - \frac{n}{2}))$$

$$O(n \text{ x } (\frac{n^2}{2} - \frac{n}{2})) = O(\frac{n^3}{2} - \frac{n^2}{2})$$

$$O(\frac{n^3}{2} - \frac{n^2}{2}) = O(\frac{n^2(n-1)}{2})$$

Except above discussed inner "for" loop that iterates through combinations, every other inner "for" loop can iterate through n number of iterations, where n is any positive integer. When such a "for" loop is considered together with the outer "for" loop, which also can iterate thorugh n number of iterations, combined time complexity becomes $O(n \text{ x } n)$. It can be simplified as below.

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

Accordingly, maximum possible time complexity of above algorithm can be either $O(\frac{n^2(n-1)}{2})$ or $O(n^2)$, depending on situation. When $n > 3$, maximum time complexity of above algorithm is $O(\frac{n^2(n-1)}{2})$. When $n = 3$, $O(\frac{n^2(n-1)}{2}) = O(n^2)$. In addition to those two cases, when positive integer n satisfies $n < 3$ condition, maximum time complexity is $O(n^2)$.

The circuit diagram corresponding to the above algorithm is depicted by figure below.

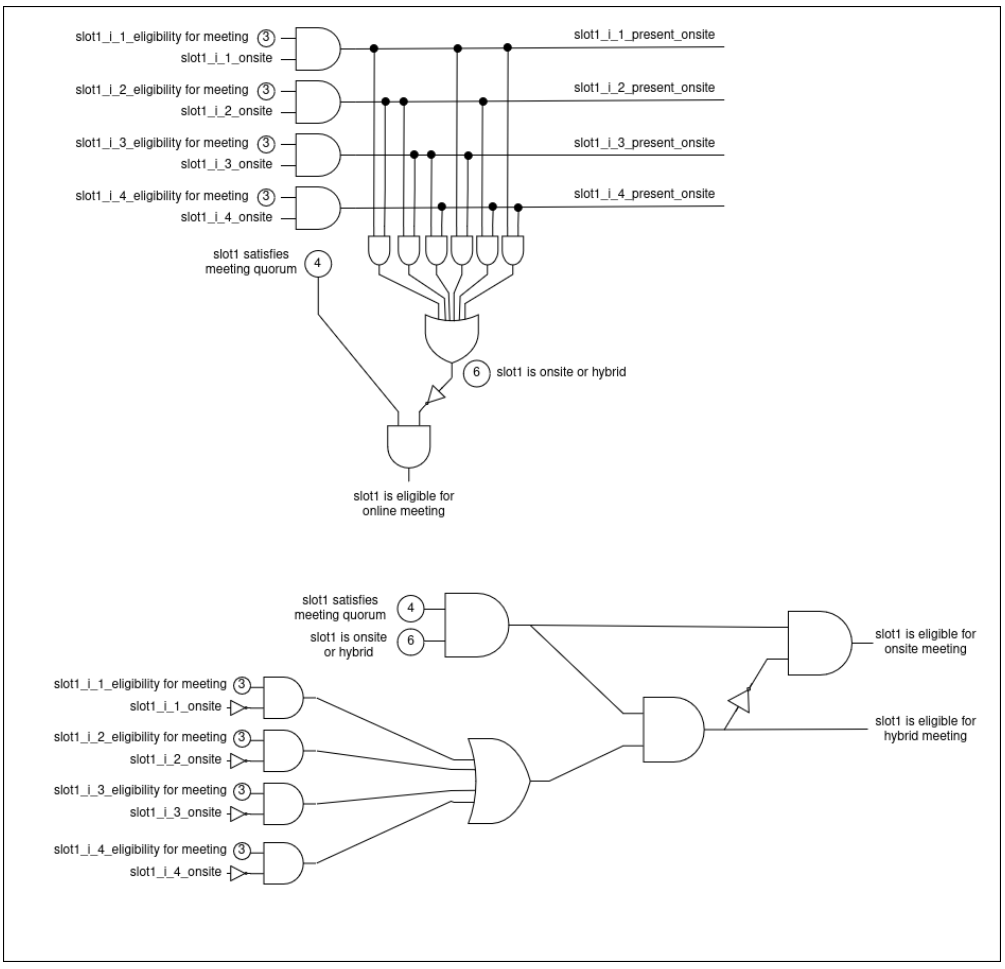

Figure 8: Circuit diagram for selecting the meeting mode

14 Complexity of the problem

14.1 Analysis based on time complexity

When above all algorithms are considered, it is observed that they don't have to be executed in parallel to each other. But, since there is an order of proceeding in operations, five algorithms are supposed to be executed from *algorithm* 1 to *algorithm* 5, sequentially. Therefore, when calculating the maximum possible time complexity of the overall problem, which has been divided into sub-problems, time complexities of those sub-problems should be compared. Following table depicts the maximum possible time complexity of each algorithm, described above.

Algorithm	**Maximum time complexity**
Algorithm 1	$O(n^2)$
Algorithm 2	$O(n^3)$
Algorithm 3	$O(n\mathrm{C}(n/2))$ [$O(n\mathrm{C}(n/2)) \approx O(2^n)$]
Algorithm 4	$O(n^2)$
Algorithm 5	$O(n^2)$ or $O(\frac{n^2(n-1)}{2})$

Table 3: Maximum time complexities of algorithms in boolean circuit mapping

Above time complexities are calculated for generating the boolean circuit dynamically, for a particular scenario, based on inputs provided. After generating the boolean circuit dynamically based on inputs provided, output can be obtained within a constant time. When above time complexities are considered, it can be observed that *algorithm* 1, *algorithm* 2, *algorithm* 4 and *algorithm* 5 have polynomial time complexities. Because, they have constant values such as 2, 3, as the power of base n. But, in *algorithm* 3, maximum possible time complexity is approximately $O(2^n)$. This is a non-polynomial time complexity. Because, 2^n increases exponentially, when value of n increases, with constant base 2. Accordingly, maximum time complexity among these algorithms is $O(2^n)$.

Due to presence of $O(2^n)$ time complexity in problem mapping, it can be concluded that our research problem is a non-polynomial (NP) time solvable problem (Pfleeger 1996). There is no specific algorithm identified to solve NP problems. Therefore, it is recommended to use an eligible heuristic or a greedy algorithm to solve such problems.

14.2 Analysis on NP-completeness of problem

When related work associated with meeting scheduling problems are considered, most of them are introduced as NP-complete problems (Bofill et al. 2022). NP-complete problems are problems belonging to both, NP and NP-hard categories (Hosch 2024). NP problems are problems that cannot be solved within a polynomial time duration, like explained earlier.

On the other hand, NP-hard problems cannot be solved during a polynomial time, as well as usually involve an optimization of variables. Graph edge coloring problem is a famous NP-hard problem. Graph edge coloring problem is, finding the number of colors required for coloring all edges of a graph, such that no two adjacent edges get the same color. Bofill et al have mapped the Business-to-Business meeting scheduling problem to edge coloring problem, for showing that it is NP-hard (Bofill et al. 2022).

Satisfiability problems such as traveling salesman problem, knapsack problem, circuit satisfiability problem, are considered as famous NP-complete problems. Therefore, it is possible to prove that any problem is NP-complete, by mapping it to one of those already known NP-complete problems (Hosch 2024). Even though our research problem which was mapped to a boolean circuit successfully as explained earlier, for analyzing the problem complexity, this problem was not mapped to any NP-complete problem. When this circuit mapping itself is considered, obviously our requirement is to find whether privacy preserved meeting is possible or no, when inputs are known. But, our requirement is not to adjust the inputs of circuit for conducting a privacy preserved meeting as the output. **Therefore, it is concluded that our problem does not belong to NP-complete category.**

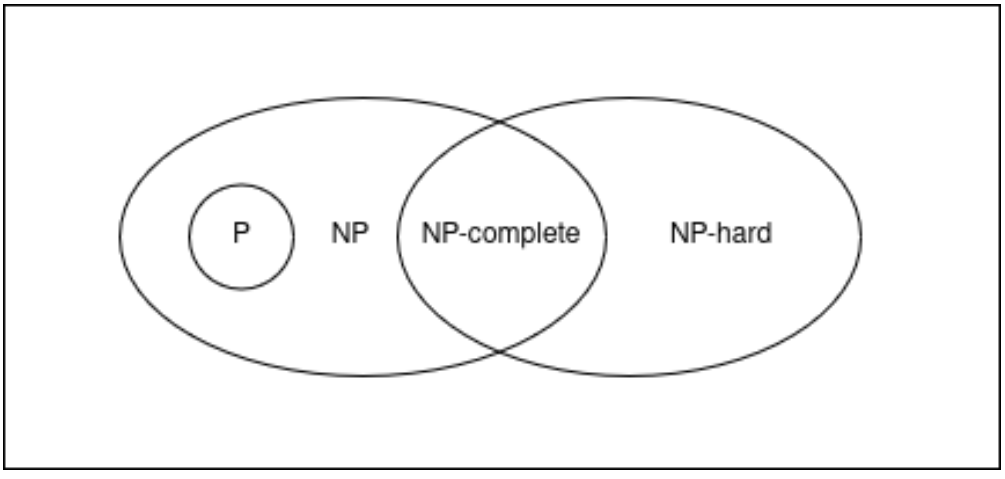

Figure 9: Problem domains based on complexity

In above diagram, NP, NP-hard and NP-complete regions represent the non-

polynomial time solvable problem categories explained earlier. P region represents the polynomial time solvable problems. Accordingly, based on our problem analysis, our research problem that was mapped to a boolean circuit, resides in NP region.

15 Implementation of boolean circuit

15.1 Choice of technology

As explained earlier, our research problem was mapped to a boolean circuit. And, that circuit was elaborated as five sequential algorithms, representing consecutive sub-sections of the entire boolean circuit. Since it is required to show that, those algorithms are convertible to a practical application in real world, a program was developed in our research.

After analyzing potentially eligible programming languages and technologies, Python was selected as the programming language for implementing the program, representing five algorithms explained ealier. Because, in our analysis, it was noticed that Python has libraries containing built-in functions, executing the behavior of logic gates. In algorithms explained above, AND, OR and NOT logic gates are utilized. Therefore, when Python is used to implement above algorithms, it is not required to simulate the behavior of logic gates by us, since there are built-in Python functions. When developing the program as proof of concept, we used hard coded inputs. When executing the program for a new scenario, those inputs can be updated by editing the Python code.

In addition, it is required to create mathematical combinations in *algorithm* 3, as explained earlier. In Python, there are libraries contaning functions that support to create such combinations. Because, creating combinations is a problem of non-polynomial time complexity.
After compiling the Python code with a compiler, execution can be done in a console environment, such as Linux terminal or the console of a suitable Integrated Development Environment (IDE).

15.2 Details on technology used for implementation

Following table depicts the the version numbers and other specific details, regarding the technologies used to implement the program, as proof of concept.

Technology	Name and version
Programming language	Python 3.7.0
Compiler	GCC 7.2.0
Library for logic gates	Sympy 1.2
Library for mathematical combinations	Itertools bundled with Python 3.7.0
Output console	Terminal of Ubuntu 22.04.4 LTS

Table 4: Technologies used to implement the proof of concept for algorithms

15.3 Overview of the Python program

This Python code has been designed to align with the definitions defined by us, earlier in this research, regarding concepts such as documents, groups of individuals, access control lists, classification of locations into three groups, time slots, meeting quorum etc. Therefore, it is recommended to be familiar with those definitions, before analyzing the Python code.

Above algorithms are designed to accept the **individuals** present in **groups** in access control lists of documents, including the agenda, as input. Therefore, in this Python program, when defining the access control lists of documents, they should be defined in terms of singleton groups (i.e. individual person), instead of aggregate groups containing multiple individuals. But, *public* group present in access control list of any document should not be resolved, since it is required to decide the eligible location for conducting the meeting. Following example depicts how access control lists should be defined in the Python program, for documents including the meeting agenda.

```
doc1 = {"public"}
doc2 = {"i_2", "i_3", "i_4", "i_5", "i_6"}
agenda = {"public", "i_1", "i_2", "i_3", "i_4", "i_5"}
```

Figure 10: Sample code snippet for defining access control lists of documents

In addition, availability of individuals in different time slots, at different locations is stored in program using a dictionary. Sample section from this dictionary is depicted in the code snippet below.

```
availability_dictionary = [
    {"person": "i_1", "slot": "slot1", "location": "onsite"},
    {"person": "i_1", "slot": "slot2", "location": "remote_private"},
    {"person": "i_2", "slot": "slot1", "location": "remote_public"}
]
```

Figure 11: Sample code snippet for availability dictionary

Further, meeting quorum also should be defined as an input variable of integer type, in the program. Following example depicts how to define it, to mention the minimum number of participants required for conducting the meeting.

```
meeting_quorum = 3
```

Figure 12: Sample code snippet for defining meeting quorum

When above mentioned inputs are defined by user, in the Python program, all required inputs are present in the code. There are five functions corresponding to five algorithms explained earlier. Each function has parameterized inputs. Some of those inputs are from inputs defined by user as explained above. In the program, these five functions are executed in sequential order, such that some outputs from initial functions are provided to later functions, as parameterized inputs. Following table depicts the name of Python function in code, corresponding to each algorithm explained earlier. In addition, output expected from execution of each function is also mentioned in the table.

#	Function name in code	Output of the function
1	`doc_analysis_validation`	Validity of each individual for meeting, by document analysis
2	`time_slot_and_participant_analysis`	Validity of each individual, in each time slot, by analyzing locations with expected privacy level for documents
3	`meeting_quorum_analysis`	Meeting quorum satisfiability of each time slot, by validated participants
4	`find_earliest_eligible_slot`	Earliest privacy preserving time slot for the meeting
5	`select_meeting_mode`	Meeting mode for each privacy preserving time slot identified

Table 5: Python functions corresponding to algorithms representing the boolean circuit (# column represents the algorithm number)

As it is described in problem analysis section, *algorithm* 3 contains an operation to create mathematical combinations. Since that operation has non-polynomial time complexity of $O(2^n)$, it has been implemented using *combinations* function of *itertools* library in Python. Because, there is no specific standard method to create combinations within polynomial time. Accordingly, it can be stated that, *itertools* library was used by us, as an existing heuristic approach to create mathematical combinations.

When the program is compiled and executed, above mentioned outputs are displayed in the console, as boolean outputs. Some examples of such outputs are depicted in figures below.

```
Participant validity by doc analysis:
{'i_2': True, 'i_4': True, 'i_6': False, 'i_1': False,
'i_5': True, 'i_3': True}
```

Figure 13: Output depicting participants validated by document analysis

In above output, *true* or $false$ states whether each individual is validated by analyzing access control lists of documents, including the agenda document as well. If an individual has no access to all the documents discussed in the meeting, particular individual will be marked as $false$.

```
Time Slot: slot1
    slot1_i_2_eligibility: True
    slot1_i_4_eligibility: True
    slot1_i_6_eligibility: False
    slot1_i_1_eligibility: False
    slot1_i_5_eligibility: False
    slot1_i_3_eligibility: True
```

Figure 14: Output depicting eligibility of each participant in a time slot, based on analysis of location with required level of privacy for documents

Above section of output displays the eligibility of each individual to attend the meeting, in each time slot. To decide the eligibility of an individual, in a certain time slot, algorithm analyzes the location of that individual, with the expected level of privacy for documents in the meeting.

```
Slot: slot1 - Quorum Satisfiability: True
Slot: slot2 - Quorum Satisfiability: True
Slot: slot3 - Quorum Satisfiability: True
Slot: slot4 - Quorum Satisfiability: False
```

Figure 15: Output depicting meeting quorum satisfiability of each time slot

Above section of output states whether each time slot satisfies the meeting quorum, by having enough number of participants at privacy preserving locations, according to the expected level of privacy for documents in the meeting.

```
Slot 1 eligibility as earliest slot: True
Slot 2 eligibility as earliest slot: False
Slot 3 eligibility as earliest slot: False
Slot 4 eligibility as earliest slot: False
```

Figure 16: Output depicting earliest privacy preserving time slot for the meeting

In above section of output, that depicts the earliest quorum satisfying time slot for meeting, only one slot is supposed to have *true* value. Because, once after identifying a quorum satisfying time slot, among all the time slots arranged in sequential order, algorithm marks all other time slots after the identified time slot as, $false$. If no quorum satisfying time slot is present, all time slots will be marked as $false$, since it is impossible to identify an earliest eligible time slot.

```
Slot1:
    Eligibility online: False
    Eligibility hybrid: False
    Eligibility onsite: True
```

Figure 17: Output depicting meeting mode for each privacy preserving time slot identified

Above section of output depicts the eligible meeting mode for conducting the meeting, in each time slot. Algorithm is designed such that more than one meeting mode cannot be activated (cannot be *true*) for a particular time slot. If a time slot is not suitable to conduct a privacy preserving meeting with equal or more number of particpants than the meeting quorum value, all three meeting modes will get $false$ for that time slot, to display that meeting should not be conducted in particular time slot.

In overall, above overview describes the inputs, outputs and behavior of the Python program, implemented as proof of concept, associated with algorithms explained earlier. In our research, those algorithms are converted to the Python code such that maximum possible time complexities identified do not get exceeded.

List of References

References

Bofill, M., Coll, C., Garcia, M., Giraldez-Cru, J., Pesant, G., Suy, J., and Villaret, M. (2022). "Constraint Solving Approaches to the Business-to-Business Meeting Scheduling Problem". In: *Journal of Artificial Intelligence Research* 74, pp. 263–301. DOI: https://doi.org/10.1613/jair.1.12670. URL: https://dl.acm.org/doi/pdf/10.1613/jair.1.12670.

Hosch, William L. (2024). "P versus NP problem — Complexity Theory & Algorithmic Solutions — Britannica — britannica.com". In: [Accessed 29-01-2025]. URL: %5Curl%7Bhttps://www.britannica.com/science/P-versus-NP-problem%7D.

Pfleeger, Charles P. (1996). "Chapter 3: Security encryption systems". In: *Security in Computing, Second Edition*. Ed. by Charles P. Pfleeger. 2nd ed. Upper Saddle River, NJ: Prentice Hall. Chap. 3, pp. 69–76. ISBN: 9780133374865.