MINISTÈRE
DE L'ÉDUCATION
NATIONALE
ET DE LA JEUNESSE
Liberté
Égalité
Fraternité

EAE SIN 3

SESSION 2023

AGRÉGATION
CONCOURS EXTERNE

Section : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR

Option : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR ET INGÉNIERIE INFORMATIQUE

CONCEPTION PRÉLIMINAIRE D'UN SYSTÈME, D'UN PROCÉDÉ OU D'UNE ORGANISATION

Durée : 6 heures

Calculatrice autorisée selon les modalités de la circulaire du 17 juin 2021 publiée au BOEN du 29 juillet 2021.

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout autre matériel électronique est rigoureusement interdit.

Il appartient au candidat de vérifier qu'il a reçu un sujet complet et correspondant à l'épreuve à laquelle il se présente.

Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence. De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.
Le fait de rendre une copie blanche est éliminatoire

Tournez la page S.V.P.

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

Concours	Section/option	Epreuve	Matière
EAE	1417A	103	1268

Ce sujet est composé de 37 pages :

- d'un dossier questionnement de la page 1 à la page 14 ;
  - présentation de la page 2 à la page 3 ;
  - partie 1 de la page 4 à la page 7 ;
  - partie 2 de la page 8 à la page 11 ;
  - partie 3 de la page 12 à la page 14 ;
- d'un dossier technique (DT) de la page 14 à la page 31 ;
- d'un dossier réponse (DR) de la page 32 à la page 37.

Conseils aux candidats :

- les trois parties du questionnement sont indépendantes ;
- un parcours attentif de l'ensemble du document est conseillé avant de composer ;
- la présentation des programmes doit respecter les mots clés du langage cible ainsi que l'indentation des structures algorithmiques ;
- les réponses doivent être présentées avec clarté, rigueur et concision.

IMPRIMERIE NATIONALE – 23 0099 – D'après documents fournis

Présentation

Flex-e-Trans est un prototype futuriste de système de transport à la demande avec des véhicules électriques et autonomes. Ce système est à mi-chemin entre les taxis individuels et les transports collectifs de masse. Comme les taxis individuels, il propose un service de porte à porte aux passagers. Comme les transports collectifs, les passagers acceptent de partager une partie de leur itinéraire avec d'autres, la contrepartie étant un coût moins élevé que les taxis individuels. Le système tire profit de toutes les avancées dans la mobilité électrique, les communications sans fil, l'internet haut débit, les transports autonomes, etc. Les véhicules roulent en conditions urbaines normales (pas de site propre), et peuvent souffrir de la congestion comme tout autre véhicule.

Le service de mobilité est exploité par un opérateur public ou privé. Cet opérateur dispose d'une flotte de véhicules électriques autonomes de deux types :

- adaptés aux Personnes à Mobilité Réduite (PMR dans la suite) ;
- véhicules de tourisme non-adaptés aux PMR.

L'opérateur dispose d'un dépôt dans lequel les véhicules peuvent être stockés. Le dépôt est également équipé de bornes de recharge des véhicules. Tous les véhicules démarrent et terminent leurs courses dans ce dépôt.

Les objectifs de Flex-e-Trans sont de :

- réduire la congestion routière en permettant le partage de trajets et l'augmentation des taux de remplissage des véhicules en circulation ;
- favoriser l'intermodalité et l'usage des transports collectifs, puisque le service de Flex-e-Trans peut servir comme un service de rabattement vers les transports en commun ;
- résoudre le problème du premier et du dernier kilomètre, qui sont parmi les raisons de la préférence de la voiture privée ;
- améliorer l'accessibilité à moindre coût pour les PMR.

Le système est composé de :

- une application Web ou mobile destinée aux passagers ;
- une application Web destinée aux administrateurs du système ;
- une application serveur permettant la communication avec les véhicules, l'optimisation des missions des véhicules, l'intégration des nouvelles demandes en temps-réel et la supervision des missions en cours.

L'opérateur dispose des ressources suivantes :

- une flotte de véhicules de deux types (adaptés aux PMR ou non) ;
- un dépôt pouvant contenir l'ensemble de la flotte, et équipé de bornes de recharge permettant de recharger en charge rapide un quart de la flotte de véhicules simultanément.

Dans un dépôt central se trouve un ensemble de véhicules avec des niveaux de charge différents. Certains sont en train d'être chargés, d'autres non. Les missions des véhicules autonomes sont composées d'un ensemble d'adresses à visiter, avec des horaires de passage estimés selon l'état du trafic courant. Ces missions sont continuellement mises à jour avant le démarrage de la mission, et éventuellement pendant l'exécution de celle-ci.

Un client (qui ne sera pas nécessairement le passager) peut à tout moment se connecter au système et demander un trajet en fournissant les informations suivantes :

- l'adresse d'origine et l'adresse de destination ;
- les informations sur le passager (PMR ou non) ;

- l'horaire de départ au plus tôt à l'adresse d'origine et l'horaire d'arrivée au plus tard à l'adresse de destination.

Un code QR qui permet l'accès du passager au véhicule lui est alors est délivré.

La figure 1 représente le système Flex-e-Trans. Les étiquettes des arêtes pleines représentent le temps de parcours courant. Les arcs en pointillés représentent l'origine et la destination du passager. L'horaire associé aux passagers au nœud de **départ** est l'horaire de départ au plus tôt. L'horaire associé au nœud d'**arrivée** représente l'horaire d'arrivée au plus tard. Les nœuds peuvent être des carrefours, ou des points de collecte et de dépose des passagers.

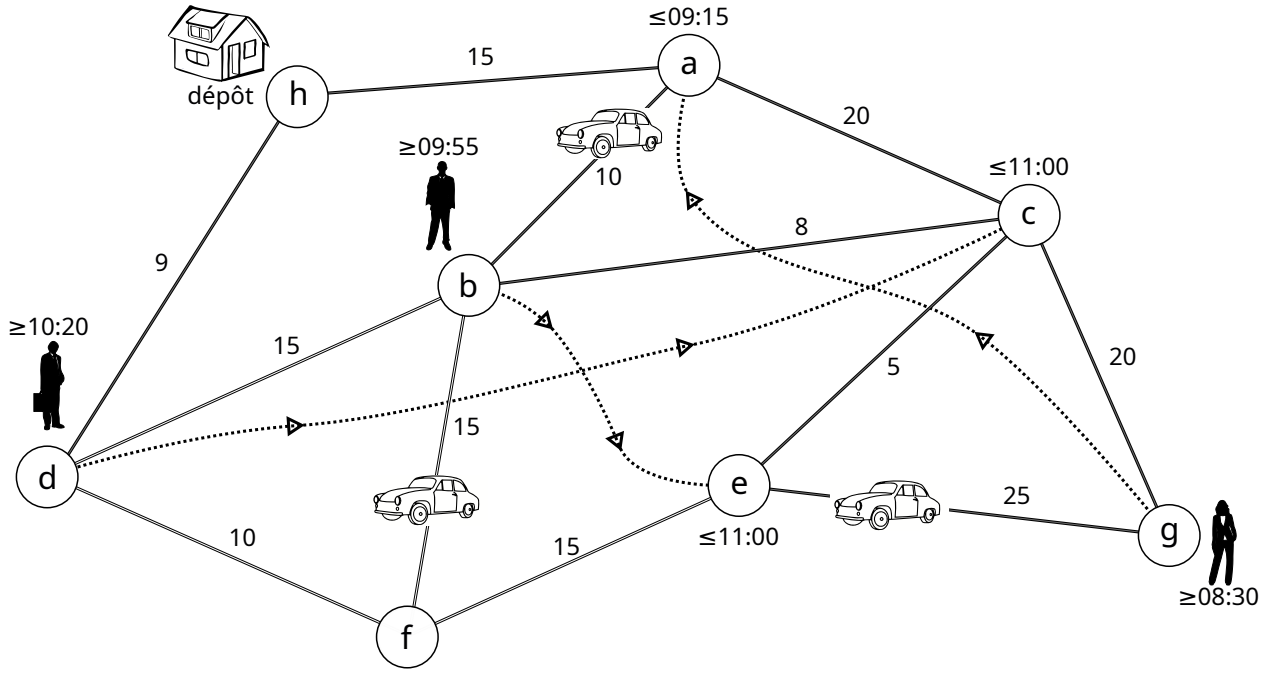


Figure 1: Système Flex-e-Trans

Le système d'optimisation a pour objectifs de prévoir et ajuster les trajets, en respectant les contraintes horaires des clients, tout en minimisant la distance parcourue par les véhicules, ainsi que le nombre de véhicules utilisés. Le système prend aussi en compte les contraintes liées à la recharge des véhicules électriques.

L'épreuve se décompose en trois parties distinctes et indépendantes :

- la partie 1 est relative à la conception du système d'information et au mode d'accès par code QR aux véhicules ;
- la partie 2 traite de certaines parties matérielles et des protocoles réseau des unités mobile ;
- la partie 3 traite de l'algorithme d'optimisation des missions des véhicules.

Partie 1 : Syst. d'information et accès véhicule

Les clients disposent d'un accès à un service Web qui permet de réserver et payer des trajets.

Pour réserver un trajet, le client doit disposer d'un compte et d'un moyen de paiement sur le site Flex-e-Trans. La création de ce compte peut être réalisée via le site Web de Flex-e-Trans. Elle permet d'obtenir un identifiant et un mot de passe.

La demande de prise en charge pour un trajet particulier se solde par la délivrance d'un code QR. C'est ce code qui permet par la suite d'accéder au véhicule. À noter que la personne qui réserve le trajet n'est pas nécessairement la personne qui voyage, le code QR pouvant être transmis / imprimé puis donné etc. : l'utilisateur qui réserve le trajet est celui qui est facturé, mais pas nécessairement celui qui utilise effectivement le service de transport.

La demande d'une réservation de trajet est réalisée en précisant :

- un point de départ, un point d'arrivée ;
- des horaires limites pour le départ (horaire au plus tôt) et pour l'arrivée (horaire au plus tard) ;
- le fait que le passager soit une personne à mobilité réduite ou non.

Le système ouvre alors une demande de réservation et propose un itinéraire prévu (il peut être amené à changer), qui peut être accepté ou refusé par le demandeur. Si le demandeur refuse le trajet, la demande est close. Dans le cas contraire, le système débite le client du prix du trajet puis fournit un code QR pour le passager, qui lui permettra par la suite d'accéder au véhicule.

La partie backend du service Web est programmée en Python. Le code exécuté sur le serveur Web est donc du code écrit en Python.

Sous-partie 1.1 : Modes de paiement et trajets

Le schéma relationnel de la figure 2 indique les relations entre quelques tables du système d'information. On y trouve par exemple les comptes clients et les réservations de trajets.

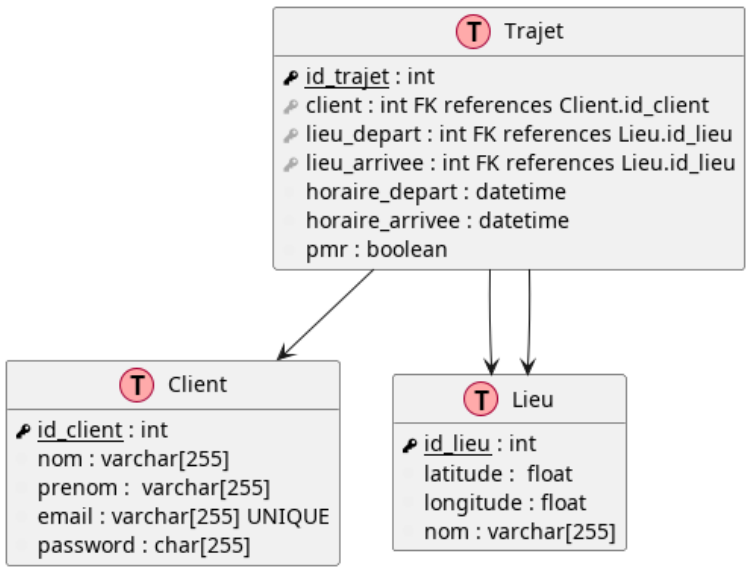


Figure 2: Schéma relationnel d'une partie du système d'information

Ajout des modes de paiement

On souhaite ajouter deux moyens de paiements :

Tournez la page S.V.P.

- un paiement par carte bleue (on doit stocker le numéro de la carte (16 chiffres), le nom du porteur (ce n'est pas nécessairement le nom du client), la date d'expiration (mois et année) et le cryptogramme de 3 chiffres (*Card Validation Value*, *CVV*) ;
- un paiement par prélèvement bancaire (on doit alors stocker l'IBAN du compte bancaire du client qui est une série de 32 symboles alphanumériques).

Q1. Proposer des modifications / des ajouts au schéma relationnel précédent (refaire un schéma similaire à celui de la figure 2) pour intégrer ces deux types de paiements possibles en prenant en compte les éléments suivants :
- Un client peut posséder plusieurs cartes bleues et / ou comptes pour prélèvement.
- Si deux personnes ont renseigné la même carte ou le même compte, alors les informations sont doublées dans le système d'information. Par exemple, si deux clients utilisent la même carte bleue, la carte en question sera stockée deux fois dans la base.

Accès à la base de données

Le document technique DT1 rappelle quelques éléments de syntaxe SQL.

Q2. Donner la requête SQL (`SELECT`) qui permet de récupérer uniquement le nom et le prénom d'un client, connaissant son email : `john.smith@yahoo.fr`

Q3. Donner la requête SQL (`SELECT`) qui permet de récupérer (uniquement) tous les couples latitude / longitude des **points de départs** des trajets que John Smith (identifié par son email `john.smith@yahoo.fr`) a **déjà réalisé** (les trajets planifiés ou en cours ne doivent donc pas être inclus).

Q4. Donner la requête SQL (`SELECT`) qui permet de récupérer toutes les informations sur le prochain (au sens de l'horaire de départ) trajet planifié par un client, à partir de son email. La requête ne devra donc donner qu'une seule réponse au maximum, contenant exactement 6 colonnes : les coordonnées géographiques et les horaires des points de départ et d'arrivée.

Afin de garder une base de code facile à maintenir, et plutôt que d'utiliser un ORM (Object-Relational Mapping), on décide de regrouper toutes les fonctions d'accès à la base de données dans une classe nommée `Database` du module `database`.

Des extraits de plusieurs modules (dont le module `database`) sont donnés dans le document technique DT2.

Q5. En s'inspirant de la méthode `Database.get_hashed_password`, donner le code de la méthode `Database.get_client_name_from_email` (prototype visible dans le DT2).

Authentification des clients

Comme indiqué dans le schéma relationnel précédent (figure 2), un client est identifié par son adresse email, et authentifié par la connaissance du mot de passe associé. Le champ `id_client` de la table `Client` est uniquement utilisé en interne, et est inconnu du client lui-même.

Lors de la création d'un compte, le nouvel utilisateur fournit une adresse email. Il reçoit alors sur cette adresse un lien Web qui le dirige vers une page lui permettant de choisir son mot de passe, d'entrer son nom et son prénom.

Ces informations sont stockées dans la table `Client` (après salage et hachage pour le mot de passe). Les documents techniques DT3 et DT4 donnent quelques rappels sur le salage et le hachage des mots de passe.

Q6. Expliquer en 10 lignes maximum pourquoi les mots de passe sont hachés avant d'être stockés dans la base de données ?

Q7. Expliquer en 10 lignes maximum pourquoi les mots de passe sont salés avant d'être hachés ?

Q8. La méthode `Database.store_password` (voir DT2) reçoit en paramètres un email et une chaîne de caractères arbitraire (le mot de passe), et doit stocker le mot de passe salé et haché dans la base. Elle renvoie `True` si l'insertion de la nouvelle donnée s'est déroulé correctement et `False` sinon. Cette fonction utilise une fonction annexe du module `outils`, nommée `hash_with_salt`. Compléter le code de la méthode et de la fonction sur le document réponse DR1.

Q9. Compléter sur le document DR2 la méthode `Database.check_password` qui indique en renvoyant un booléen si un couple login / mot de passe est valide ou non.

Sous-partie 1.2 : Génération du code QR

Afin que le système d'accueil des passagers dans les véhicules dépende le moins possible de l'accès réseau des véhicules au système d'information, le code QR que chaque passager doit présenter pour utiliser un véhicule contient **toutes** les informations nécessaires sur le trajet et l'identité du client : points de départ et d'arrivée, horaires, identification du client (on rappelle que le client n'est pas nécessairement le voyageur) et de la course.

Ce code (et lui seul) doit pouvoir être utilisé comme preuve par un client pour une réclamation éventuelle (problème d'horaires, véhicule non présent sur le lieu prévu etc.).

Le caractère *sécurisé* du code QR est important :

- un utilisateur ne doit pas pouvoir forger un faux code QR, et prétendre ainsi qu'il a réservé un trajet ;
- un utilisateur ne doit pas pouvoir modifier les horaires, lieux ou tout autre chose dans son code QR, sans que ce soit immédiatement décelable à la lecture du code ;
- lorsque l'utilisateur se présente devant le véhicule, ce dernier doit être en mesure de vérifier que le code QR est correct, même s'il n'a aucun accès réseau, et ne peut pas joindre sa centrale.

Afin de satisfaire ces contraintes, les codes QR contiennent une signature numérique, basée sur l'algorithme HMAC SHA-256 dont les grandes lignes sont données dans le document DT5.

Pour stocker une séquence d'octets dans un code QR, l'usage est d'encoder les données en *base 45*, puis d'utiliser la fonctionnalité de stockage d'une séquence alphanumérique (contenant justement 45 symboles) dans le code QR. Les étapes de génération du code QR, à partir des données sur le trajet sont donc :

1. Organiser les données selon un format binaire compact.
2. Calculer la signature numérique de la séquence d'octets obtenue.
3. Encoder en *base 45* les données signées (séquence d'octets + signature).
4. Générer le code QR à partir de la chaîne contenant les données en *base 45*, préfixée par un entête qui permet d'identifier le type de code

Le document technique DT6 détaille le fonctionnement de l'encodage *base 45*.

Tournez la page S.V.P.

On dispose d'un certain nombre de fonctions Python développées dans le but de générer le code QR et de le vérifier.

Les fonctions et classes développées sont données dans le document technique DT2.

La classe `CourseClient` permet de décrire une course particulière pour un client particulier. Elle contient les informations qui seront présentes dans le code QR : points de départ et d'arrivée, horaires etc.

Les données à encoder doivent être mises sous forme binaire compacte (pour limiter la taille du code QR). Ceci est réalisé par la méthode `CourseClient.to_bytes`.

Q10. Dans le document réponse DR3 compléter la méthode de classe `CourseClient.from_bytes` qui réalise l'opération inverse.

Q11. La fonction `sign` du module `outils` permet d'obtenir un bloc signé à partir d'un bloc de données. Dans la fonction `sign`, pourquoi la valeur de `n` est-elle comparée à `0xffff` ?

Q12. Sur le document réponse DR4, compléter la fonction `check_sign` du module `outils` qui prend en paramètre des données signées, et indique en renvoyant un booléen si la signature est valide.

La fonction `base45_encode` du module `outils` prend en paramètre un objet de type `bytes`, et l'encode en *base 45*.

La spécification de l'encodage *base 45* est donnée dans le document technique DT6.

Q13. Sur le document réponse DR5, compléter la fonction `base45_encode` et les fonctions annexes utilisées.

Q14. Écrire la méthode `CourseClient.create_qr_code` dont le prototype est donné dans le document DT2, qui renvoie l'image du code QR. Toutes les fonctions mentionnées dans le document DT2 sont utilisables.

Q15. Écrire la méthode de classe `CourseClient.read_qr_code` dont le prototype est donné dans le document DT2. Cette méthode prend en paramètre l'image du code QR, et renvoie :

- `None` si le code QR n'est pas valide (signature erronée) ou n'est pas reconnu ;
- un objet de type `CourseClient` sinon.

Toutes les fonctions mentionnées dans le document DT2 sont utilisables.

Partie 2 : Équipement et communications

Les systèmes de transport intelligents (ITS pour *Intelligent Transportation Systems*) font l'objet de plusieurs normalisations, internationales, européennes ou nationales. En Europe, c'est l'Institut européen des normes de télécommunications (ETSI pour *European Telecommunications Standards Institute*) qui est en charge de cette normalisation, définissant les communications, entre les véhicules eux mêmes, les différentes stations (unités de bord de route), Internet etc.

Les différentes technologies mises en œuvre tiennent compte du fait que :

- les stations sont mobiles, modifiant en continue la topologie du réseau ;
- les couches basses de communication peuvent être très diverses (Bluetooth, Wifi, GSM...) ;
- on trouve aussi une grande diversité dans les terminaux (véhicule lui-même, station de bord de route, smartphone, centrale...) et les applications (routage, régulation du traffic, conduite autonome, système d'appel d'urgence...).

Sous-partie 2.1 : Système eCall

Les véhicules Flex-e-Trans sont équipés du système *eCall* qui permet à un véhicule, en cas d'accident, de prévenir les secours (communication avec le numéro de secours européen 112). Les véhicules disposent donc d'un module de communication sur le réseau mobile, pour appeler le 112, et peuvent transmettre de la voix, et des données. Celles-ci doivent en particulier contenir la position du véhicule en détresse, qui est acquise grâce au système de positionnement par satellite (GNSS) du véhicule. Le système de positionnement en question communique avec les autres organes du véhicule, dont le système *eCall*, par le biais de trames au format NMEA 0183. Le format de ces trames est détaillé dans le document technique DT7.

Q16. Le type d'une trame NMEA 0183 est donné par 3 caractères contenus dans la trame. Parmi les types présentés dans le document DT7, quel est celui qui sera utilisé pour récupérer la position du véhicule ?

Q17. Quels sont les deux caractères manquants à la fin de la trame donnée ci-dessous ? Une table ASCII est donnée dans le document technique DT8.

```
$GPGSA,A,3,06,07,16,20,14,,,24,,07,,,2.5,1.3,2.1*
```

Le microcontrôleur qui gère le système eCall est programmé en langage C. Il a pour fonction de détecter, par le biais de différents capteurs (coussin auto-gonflant etc.) un éventuel accident, et prévient les secours dans ce cas. Parmi les tâches qui lui incombent, figurent la lecture des trames NMEA 0183 qui émanent du module GPS (lecture sur le port série) et le décodage de ces trames, de manière à accéder à la latitude, à la longitude, et à l'horodatage.

La structure capable de stocker ces informations est définie ainsi :

```
typedef struct {
  float latitude, longitude;
  int heures, minutes;
  float secondes;
} position;
```

Tournez la page S.V.P.

Q18. L'objectif est maintenant, étant donnée une trame NMEA 0183 (chaîne de caractères), de l'analyser et de renvoyer une structure contenant les informations d'horodatage et de position. Une partie du code est donnée dans le document DR6. Compléter les différentes fonctions, dont la fonction C `int get_position(char * nmea_frame, position * pos)` qui prend en paramètres une trame NMEA 0183 et un pointeur vers une structure `position`, et remplit la structure avec les informations si la trame NMEA 0183 le permet. En cas de succès, la fonction devra renvoyer 1. Si la trame NMEA 0183 ne permet pas de connaître ces informations (si ce n'est pas le bon type de trame par exemple) ou si la trame contient une erreur (lecture impossible, checksum incorrect…), la fonction devra renvoyer 0. Il sera admis que si le checksum est correct, alors les différents éléments de la trame sont corrects : il sera par exemple inutile de vérifier que la latitude contient le bon nombre de chiffres.

Sous-partie 2.2 : Couche réseau et transports - GeoNetworking

Les couches réseau et transport d'un système de transport intelligent contiennent les protocoles historiques d'Internet, TCP / UDP et IPv6, mais aussi un protocole réseau plus spécifique appelé *GeoNetworking*. Ce protocole permet des communications réseaux sans fil ad-hoc entre les différents terminaux, et favorise la diffusion des messages en utilisant la position géographique des nœuds du réseau. Dans ce mode de communication, chaque terminal communique directement avec d'autres terminaux sans l'intermédiaire de routeur dédié (chaque nœud peut participer à la diffusion du message à ses voisins).

GeoNetworking propose plusieurs mode de diffusion de l'information :

- *GeoUnicast* : on cible un destinataire avec son adresse (figure 3) ;
- *GeoBroadcast* : on cible tous les nœuds d'une zone géographique délimitée (figure 4) ;
- *GeoAnycast* : on cible un des nœuds d'une zone géographique délimitée ;
- *TSB* (*Topologically Scoped Broadcast*): diffusion aux n plus proches voisins (figure 5).

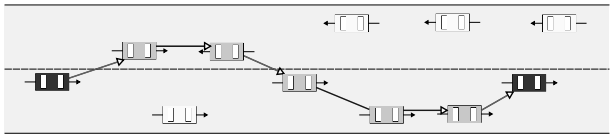


Figure 3: Diffusion de type *GeoUnicast*

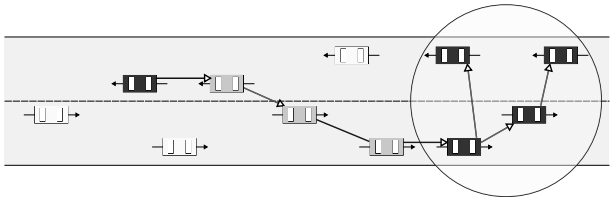


Figure 4: Diffusion de type *GeoBroadcast*

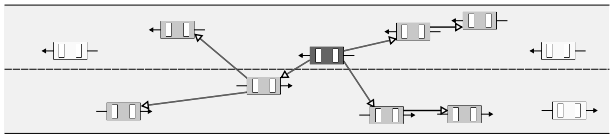


Figure 5: Diffusion de type *TSB*

Adressage GeoNetworking

GeoNetworking propose un système d'adressage qui lui est propre, avec des adresses de 8 octets. Ce système est décrit dans le document technique DT9.

Q19. On donne le début d'une adresse en notation hexadécimale pointée : `15.0C.xx.xx.xx.xx.xx.xx`
Indiquer le type de station (véhicule ou bien unité de bord de route ? feu de signalisation, vélo, voiture…?), le caractère privé ou public de la station, ainsi que le nom du pays correspondant à la station.

Vecteurs de Position

À plusieurs niveaux, il est nécessaire de décrire une position géographique :

1. une station peut informer les stations environnantes de sa position, en utilisant un *Long Position Vector* ou un *Short Position Vector*
2. une zone peut être ciblée, en donnant un point ou une aire géographique (possibilité offerte dans l'entête des trames GeoNetworking)

La structure des Long Position Vectors est donnée dans le document technique DT10.

Q20. Le champs `TST` propose un horodatage des données de position. Du fait du nombre de bits utilisés pour encoder ce champs, la valeur `TST` cycle et repasse régulièrement par la valeur 0. Avec quelle période, exprimée en jours, ce phénomène se produit-il ?

Trames GeoNetworking

L'entête des trames GeoNetworking est décrit dans le document technique DT11.

Q21. En supposant que l'unité est la seconde, si le champ `LT` de l'entête code un nombre entier sur un octet, quelle est la durée maximum représentable (`vmax`) et quelle est la plus petite durée non nulle représentable (`vmin`) ?

Afin d'avoir une meilleure résolution sur les petites valeurs, `LT` utilise un encodage non linéaire qui n'est pas détaillé dans les documents techniques, mais uniquement ci-après :

- les 6 bits de poids fort de `LT` représentent un nombre entier nommé *multiplieur*
- les 2 bits de poids faible de `LT` encodent la *base* :

code base	valeur base
0	50 ms
1	1 s
2	10 s
3	100 s

La valeur encodée est obtenue en faisant le produit entre *multiplieur* et *valeur base*. Par exemple, le code `LT`=14 (`0b00001110`) correspond à un *multiplieur* égal à 3 (`0b000011`) et à une *valeur base* égale à 10 s (*code base* égale à 2, `0b10`). La durée encodée par `LT`=14 correspond donc à une durée de 30 secondes ($3 \times 10\,\mathrm{s}$).

Q22. En utilisant ce codage, quelles sont les nouvelles valeurs de `vmin` et `vmax` ? Quels octets (donner des nombres entiers en base 10) permettent de représenter ces deux valeurs extrêmes ?

Tournez la page S.V.P.

Algorithmes de routage

Les algorithmes de routage réseau *GeoNetworking* nécessitent, pour deux points $M(x, y)$ et $C(x_c, y_c)$, l'utilisation d'une fonction F :

$$\begin{cases} \mathbb{R}^2 \to & \mathbb{R} \\ (x, y) & \mapsto \quad F(x, y) \end{cases}$$

telle que :

- $F(x, y) = 1$ si $M(x, y) = C$
- $F(x, y) > 0$ si $M(x, y)$ appartient au disque ouvert de centre C et de rayon r
- $F(x, y) = 0$ si $M(x, y)$ appartient au cercle de centre C et de rayon r
- $F(x, y) < 0$ si $M(x, y)$ n'appartient pas au disque fermé de centre C et de rayon r

Q23. Proposer une fonction F qui satisfait ces conditions.

L'algorithme de routage *Simple Geobroadcast Forwarding Algorithm* est donné dans le document technique DT12. Il est accompagné d'un algorithme nommé *Greedy Forwarding algorithm*.

Q24. Sur le document réponse DR7, sont représentés différents nœuds d'un réseau GeoNetworking. La distance entre chaque nœud est donnée par un tableau du document DR7. Une trame GeoBroadcast (figure 4) est émise depuis le nœud `LPV`, à destination d'une zone de rayon 3, centrée autour de `A`. Quel sera le trajet suivi par la trame pour atteindre le premier nœud destination si elle est routée en utilisant les algorithmes du document DT12, en supposant que le voisinage correspond à une distance de 4 ? Autrement dit, `(i.IS_NEIGHBOUR)` est vrai si le point `i` est à une distance inférieure ou égale à 4 du point considéré. Indiquer clairement la liste des nœuds par lesquels l'information transite et expliquer la construction du tracé.

Q25. La solution obtenue avec les algorithmes du DT12 est-elle optimale en nombre de sauts ? Si ce n'est pas le cas, trouver un trajet optimal de `LPV` à sa destination.

Q26. Dans le document DT12, l'algorithme nommé `GF ALGORITHM` est qualifié de glouton (*greedy* en anglais). Pour quelle raison (texte libre, moins de 10 lignes) ?

Q27. Pour quelle(s) raison(s) (texte libre, moins de 10 lignes) le protocole IP n'est-il pas suffisant pour assurer les services de la couche réseau et en quoi le protocole GeoNetworking permet-il de régler ce problème ?

Partie 3 : Système d'optimisation des missions

Cette partie traite de la création des missions des véhicules, et de leur suivi et mise à jour, à partir des requêtes de déplacement des passagers et de l'état actuel de la flotte de véhicules.

Le système d'optimisation a pour objectifs de :

- minimiser le nombre de requêtes rejetées pour violation des contraintes temporelles (impossibilité de respecter l'horaire de départ au plus tôt à l'adresse d'origine ou l'horaire d'arrivée au plus tard à l'adresse de destination) ;
- minimiser le nombre de véhicules mobilisés pour desservir les passagers ;
- minimiser les temps de parcours de l'ensemble des véhicules ;
- minimiser les distances totales parcourues par tous les véhicules ;
- minimiser le temps d'attente des passagers (le temps d'attente est égal à la différence entre l'horaire de départ au plus tôt et l'horaire de passage effectif du véhicule à l'adresse d'origine).

Le système doit également pouvoir réagir à des événements exceptionnels :

- la congestion qui peut violer les contraintes temporelles des passagers :
- l'annulation d'une demande ou l'absence d'un passager à l'adresse d'origine ;
- la panne d'un véhicule.

Le système d'optimisation dispose d'une représentation du réseau routier sous la forme d'un graphe $G(V, A)$ composé de $V = \{v_i\}, i = \{0, \dots, N\}$ (le nœud v_0 est le dépôt) et d'un ensemble d'arcs $A = \{(v_i, v_j) | v_i \in V, v_j \in V, i \neq j\}$. Un arc entre deux nœuds représente une route directe les reliant. Les arcs du réseau ont une distance statique d_{ij}, et un temps de parcours courant dynamique t_{ij}. Ce temps de parcours est calculé sur la base d'un réseau fluide par défaut, mais est mis à jour avec les temps de parcours réels.

Q28. Dessiner un graphe représentant le réseau de transport selon la description précédente. Les nœuds de ce graphe seront les intersections et les points de collecte et de dépose des passagers.

Q29. Donner en pseudo-code un algorithme permettant de calculer le chemin le plus rapide entre un nœud particulier $v^\star$ et tous les autres nœuds du graphe, suivant le temps de parcours courant. Quelle est la complexité temporelle de l'algorithme proposé ?

L'algorithme permettant d'insérer une nouvelle requête (collecte et dépose d'un passager) dans les missions des véhicules) est nommé `insertion_missions`.

Cet algorithme utilise une fonction nommée `insertion(r,veh)` qui insère la requête `r` d'un nouveau client dans le plan du véhicule particulier `veh`, et renvoie le temps de parcours additionnel pour `veh` suite à l'insertion de cette requête. La fonction `insertion(r,veh)` renvoie l'infini si la requête ne peut pas être insérée sans violer les contraintes temporelles, ou si le véhicule est inadapté (véhicule non adapté pour une personne à mobilité réduite par exemple). Dans la suite, on supposera la pré-existence de cette fonction `insertion(r,veh)`.

Lorsqu'un passager demande à être transporté d'une origine à une destination, l'ensemble des véhicules (ceux au dépôt comme ceux en circulation) est inspecté par l'algorithme `insertion_missions` : À l'aide de la fonction `insertion`, chaque véhicule essaie d'insérer le point de départ, puis le point de destination dans son parcours. Les itinéraires entre chaque couple de nœuds de la mission sont calculés suivant le chemin le plus rapide. Si l'insertion est possible sans violer les contraintes temporelles du nouveau passager, ni celles des éventuels passagers déjà planifiés ou à bord, le véhicule calcule le détour résultant de l'insertion du nouveau

Tournez la page S.V.P.

passager. Le véhicule avec le moins de détour est choisi pour transporter le nouveau passager. Cette heuristique a l'avantage d'afficher un temps d'exécution très rapide, condition nécessaire aux systèmes temps-réel.

Q30. Donner en pseudo-code l'algorithme de `insertion_missions` permettant le traitement d'une nouvelle requête passager selon l'heuristique décrite plus haut. L'algorithme résoudra l'insertion d'une seule requête reçue. On pourra supposer la pré-existence de la fonction `insertion(r,veh)` et donc l'utiliser dans le pseudo-code.

L'heuristique d'insertion `insertion_missions` est rapide mais « myope ». Cela signifie que l'apparition ultérieure d'une requête de passager peut rendre la précédente insertion sous-optimale.

Q31. Proposer une idée de modification de l'algorithme `insertion_missions`, qui serait plus efficace selon vous en limitant sa myopie (argumenter).

Pendant l'exécution des missions des véhicules, un écart peut être observé entre le plan et la réalité du terrain. Cet écart peut être plus ou moins important selon que cela rend invalides les missions ou non (violation des contraintes temporelles des véhicules). Trois cas sont identifiés, exigeant potentiellement un recalcul de certaines missions :

1. une congestion (augmentation des temps de parcours) ;
2. une panne d'un véhicule autonome ;
3. une annulation de requête.

Q32. Décrire (en texte libre, moins de 10 lignes) une solution permettant de déterminer si un re-calcul de missions est nécessaire ou non dans chacun des trois cas précédents.

Q33. Proposer une solution (description en texte libre, moins de 10 lignes) permettant de recalculer les missions en cas de congestion. On ne considère que les requêtes qui ne sont pas en cours d'exécution (i.e. on ne considère pas les passagers à bord des véhicules).

Q34. Proposer une solution (description en texte libre, en moins de 10 lignes) permettant de recalculer les missions en cas de panne.

Q35. Proposer une solution (description en texte libre, en moins de 10 lignes) permettant de recalculer les missions en cas d'annulation de requête.

Supposons que l'on dispose de données historique de trafic associant à chaque arc un minimum et un maximum de temps de parcours observé.

Q36. Proposer une manière d'intégrer cette information dans l'algorithme `insertion(r, veh)`, pour minimiser les recalculs d'itinéraires à cause de la congestion (description en texte libre en moins de 10 lignes).

L'utilisation de véhicules autonomes a l'avantage d'éviter les coûts associés aux conducteurs. Il est donc envisageable que des véhicules attendent, non pas dans le dépôt, mais dans des places de stationnement quelque part dans la ville.

Q37. Comment peut-on utiliser cette possibilité pour améliorer l'efficacité du système d'optimisation ?

La recharge des véhicules est une question importante dans un système d'électromobilité.

Q38. Proposer au moins deux idées pour optimiser l'autonomie énergétique des véhicules. Quels véhicules recharger au dépôt et quand ?

Considérons maintenant qu'il est possible de recharger les véhicules, non pas au dépôt mais ailleurs sur le réseau en installant des bornes de recharge un peu partout.

Q39. Proposer une idée pour le positionnement de ces bornes

Documents techniques

DT1 : Exemples de requêtes SQL sur une base de données fictive

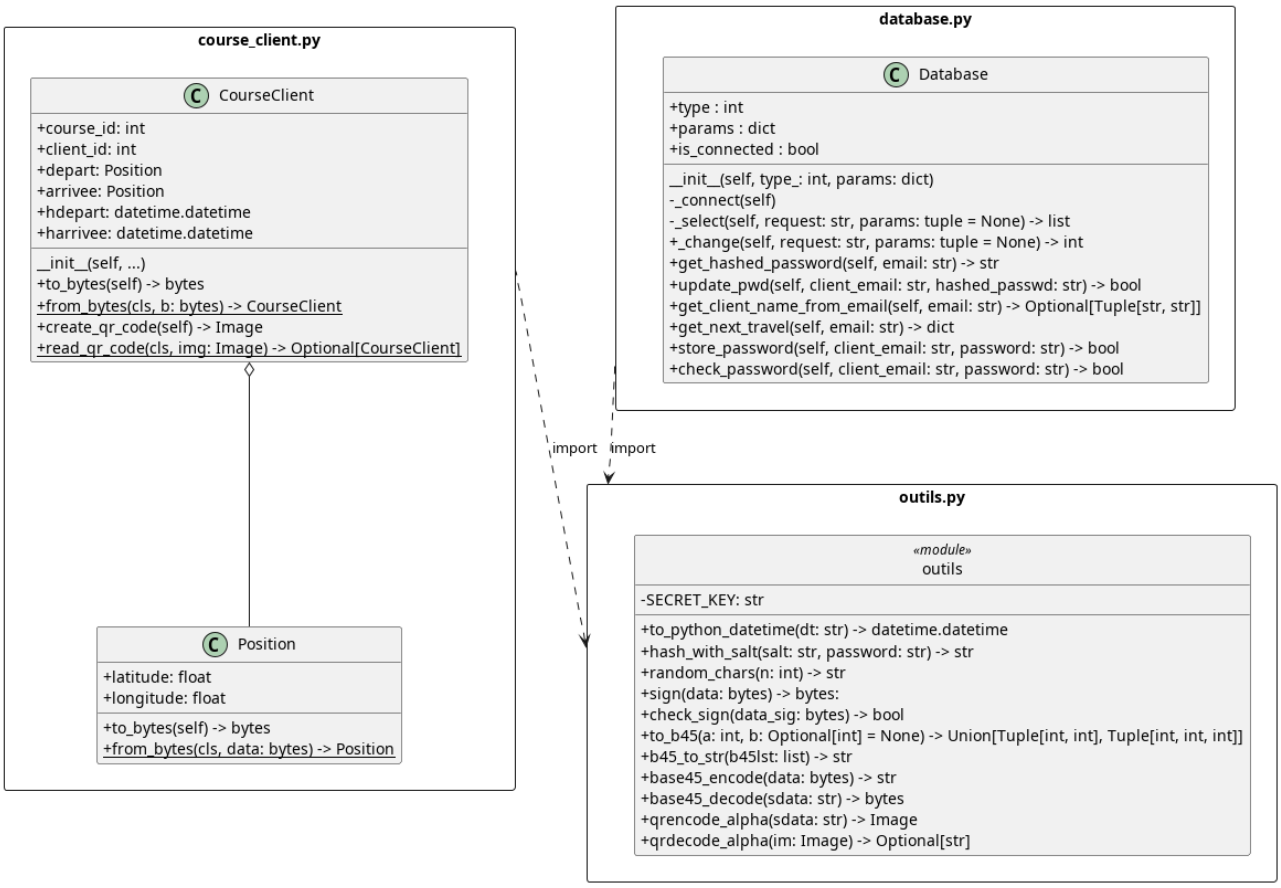
Cinq prochaines commandes à livrer :

```
SELECT commandes.total, clients.nom
   FROM commandes JOIN clients ON clients.id = commandes.id_client
   WHERE commandes.date_livraison > NOW()
   ORDER BY commandes.date_livraison
   LIMIT 5
```

Montant total des commandes pour chaque client :

```
SELECT clients.nom, sum(commandes.total)
   FROM commandes, clients
   WHERE clients.id = commandes.id_client
   GROUP BY clients.id
```

DT2 : modules `database.py`, `course_client.py` et `outils.py`



module `database.py`

```
class DatabaseValueError(Exception): pass

class Database:
    """
    Utilisation de la base de données Flex-e-trans,
    """
    def __init__(self, type_: int, params: dict):
        """
        :param type_: 0 pour sqlite, 1 pour mariadb
        :param params: dict. contenant les informations de connection
            pour sqlite3
               {'filename' : <nom_fichier> }
            pour mariadb
               {'host': <host>, 'database': <dbname>, 'username': <user>,
                'password': <password>}
        """
        self.type: int = type_
        self.params: dict = params
        self.is_connected: bool = False

    def _connect(self):
        ...
        self.is_connected = True

    def _select(self, request: str, params: tuple=None) -> list:
        " Méthode utilisée pour l'exécution d'une requête SELECT "
        if not self.is_connected:
            self._connect()
        cursor = self.conn.cursor()
        if params is not None:
            cursor.execute(request, params)
        else:
            cursor.execute(request)
        rows = cursor.fetchall()
        result = list(rows)
        cursor.close()
        return result

    def _change(self, request: str, params: tuple = None) -> int:
        """
        Méthode utilisée pour l'exécution d'une requête
        UPDATE, INSERT, DELETE. Renvoie le nombre de lignes
        modifiées par la requête
        """
        ... # Code non donnée ici
        return rowcount
```

```
    def get_hashed_password(self, email: str) -> str:
        """
        Récupération du mot de passe haché stocké dans la base à partir
        de l'email
        """
        req = "SELECT password FROM Client where email=?"
        res = self._select(req, (email,))
        if not res:
            raise DatabaseValueError("Utilisateur inconnu")
        if len(res) > 1:
            raise DatabaseValueError("Utilisateur en double")
        return res[0][0] # Renvoie le 1er champ de la 1ère ligne

    def update_pwd(self, client_email: str, hashed_passwd: str) -> bool:
        """
        Met à jour le champs password associé à un email.
        Renvoie True si la mise a jour a pu être faite.
        """
        req = "UPDATE Client SET password=? WHERE email=?"
        r = self._change(req, (hashed_passwd, client_email))
        return r == 1


    def get_client_name_from_email(self, email: str) \
            -> Optional[Tuple[str, str]]:
        """
        Renvoie le nom et le prénom d'un client à partir de son email.
        Si le client n'est pas trouvé, renvoie None
        >>> db.get_client_name_from_email("john.smith@yahoo.fr")
        ("Smith", "John")
        """
        ...

    def get_next_travel(self, email: str) -> dict:
        ...

    def store_password(self, client_email: str, password: str) -> bool:
        ...

    def check_password(self, client_email: str, password: str) -> bool:
        ...
```


Tournez la page S.V.P.

module `course_client.py`

```
from typing import Optional, Union, Tuple
import datetime
import struct
from dataclasses import dataclass
from PIL import Image
import outils

class CourseClient:

    def __init__(self, course: int, client: int, pos_depart: Position,
                 heure_depart: Position, pos_arrivee: datetime.datetime,
                 heure_arrivee: datetime.datetime):
        self.course_id: int = course
        self.client_id: int = client
        self.depart: Position = pos_depart
        self.arrivee: Position  = pos_arrivee
        self.hdepart: datetime.datetime = heure_depart
        self.harrivee: datetime.datetime = heure_arrivee

    def to_bytes(self) -> bytes:
        """ renvoie une forme binaire compacte de `CourseClient` """
        # int 4 octets pour course_id + int 4 octets pour client_id
        cc_id = struct.pack(">ii", self.course_id, self.client_id)
        # 4 + 4 octets pour depart
        depart = self.depart.to_bytes()
        # 4 + 4 octets pour arrivee
        arrivee = self.arrivee.to_bytes()
        # 2 octets pour year, month, day, hour, minute
        dep_heure = struct.pack(">hhhhh",
                                self.hdepart.year, self.hdepart.month,
                                self.hdepart.day, self.hdepart.hour,
                                self.hdepart.minute)
        # 2 octets for year, month, day, hour, minute
        arr_heure= struct.pack(">hhhhh",
                                self.harrivee.year, self.harrivee.month,
                                self.harrivee.day, self.harrivee.hour,
                                self.harrivee.minute)

        return cc_id + depart + dep_heure + arrivee + arr_heure

    @classmethod
    def from_bytes(cls, b: bytes) -> "CourseClient":
        ...
    def create_qr_code(self) -> Image:
        """ Crée l'image d'un code QR signé à partir des informations
        d'un trajet (CourseClient) """
        ...
```

```
    # suite classe CourseClient
    @classmethod
    def read_qr_code(cls, img: Image) -> Optional["CourseClient"]:
        """ Si l'image du code QR est valide, et que le code QR est
        correctement signé, renvoie l'objet CourseClient contenu
        dans le code QR. Sinon, renvoie None
        """
        ...

@dataclass
class Position:
    latitude: float
    longitude: float

    def to_bytes(self) -> bytes:
        # 4 octets (IEE754 single) pour latitude, 4 octets pout longitude
        return struct.pack(">ff", self.latitude, self.longitude)

    @classmethod
    def from_bytes(cls, data: bytes) -> "Position":
        lat, long = struct.unpack(">ff", data)
        return cls(lat, long)
```

module `outils.py`

```
import random
import hashlib
import datetime
import hmac
import string
from PIL import Image
from typing import Optional, Union, Tuple

SECRET_KEY = bytes.fromhex("043372332e372272153e352f3d2b6760")

def to_python_datetime(dt: str) -> datetime.datetime:
    """ Convertit une date issue de la base de donnée en objet
    datetime.datetime de Python
    >>> to_python_datetime("2023-06-20 10:30:00")
    datetime.datetime(2023, 6, 20, 10, 30, 0)
    """
    dtformat = "%Y-%m-%d %H:%M:%S"
    return datetime.datetime.strptime(dt, dtformat)

def random_chars(n: int) -> str:
    """ Renvoie une chaîne de n caractères pseudo-aléatoires
    (lettre ou chiffres) """
    chars = string.ascii_letters + string.digits
    lst = [random.choice(chars) for _ in range(n)]
    return "".join(lst)
```


Tournez la page S.V.P.

```
def sign(data: bytes) -> bytes:
    """ Renvoie les données signées (données + signature) au  format :
    - 1 octet pour le type de signature
    - 2 octets pour la taille des données
    - X octets pour les données
    - la signature
    """
    sig_type = 1 # Évolution possible vers d'autres signatures
    n = len(data)
    if n > 0xffff:
        raise ValueError("Too large data")
    sig = hmac.HMAC(SECRET_KEY, msg=data, digestmod=hashlib.sha256).digest()
    return bytes([sig_type, n // 256, n % 256]) + data + sig

def hash_with_salt(salt: str, password: str) -> str:
    ...

def check_sign(data_sig: bytes) -> bool:
    ...

def to_b45(a: int, b: Optional[int] = None) \
      -> Union[Tuple[int, int], Tuple[int, int, int]]:
    ...

def b45_to_str(b45lst: list) -> str:
    ...

def base45_encode(data: bytes) -> str:
    ...

def base45_decode(sdata: str) -> bytes:
    " Décode une chaîne Base45 en séquence d'octets "
    # Fonction non détaillée ici
    ...

def qrencode_alpha(sdata: str) -> Image:
    " Renvoie l'image d'un code QR à partir d'une chaîne Base45 "
    # Fonction non détaillée ici
    ...

def qrdecode_alpha(im: Image) -> Optional[str]:
    """ Décode l'image du code QR.
    Renvoie None si l'image n'est pas celle d'un code QR
    alphanumérique (mode 2). Renvoie la chaîne Base45 sinon"""
    # Fonction non détaillée ici
    ...
```

DT3 : Fonction de hachage

Extrait de https://fr.wikipedia.org/wiki/Fonction_de_hachage consulté le 15/09/21

On nomme fonction de hachage, de l'anglais hash function (hash : pagaille, désordre, recouper et mélanger) par analogie avec la cuisine, une fonction particulière qui, à partir d'une donnée fournie en entrée, calcule une empreinte numérique servant à identifier rapidement la donnée initiale, au même titre qu'une signature pour identifier une personne. Les fonctions de hachage sont utilisées en informatique et en cryptographie notamment pour reconnaître rapidement des fichiers ou des mots de passe.

Principe général

Une fonction de hachage est typiquement une fonction qui, pour un ensemble de très grande taille (théoriquement infini) et de nature très diversifiée, va renvoyer des résultats aux spécifications précises (en général des chaînes de caractères de taille limitée ou fixe) optimisées pour des applications particulières. Les chaînes permettent d'établir des relations (égalité, égalité probable, non-égalité, ordre…) entre les objets de départ sans accéder directement à ces derniers, en général soit pour des questions d'optimisation (la taille des objets de départ nuit aux performances), soit pour des questions de confidentialité.

Autrement dit : à 1 fichier (ou à 1 mot) va correspondre une signature unique (le résultat de la fonction de hachage, soit le hash).

En termes très concrets, on peut voir une fonction de hachage (non cryptographique) comme un moyen de replier l'espace de données que l'on suppose potentiellement très grand et très peu rempli pour le faire entrer dans la mémoire de l'ordinateur. En revanche, une fonction de hachage cryptographique est ce que l'on appelle une fonction à sens unique, ce qui veut dire que le calcul de la fonction de hachage est facile et rapide tandis que le calcul de sa fonction inverse est infaisable par calcul et donc non calculable en pratique. Grâce à la valeur de hachage (le hash), on peut discriminer deux objets apparemment proches, ce qui peut être utilisé pour garantir l'intégrité des objets, autrement dit leur non-modification par une erreur ou un acteur malveillant.

Les algorithmes SHA-1 (Secure Hash Algorithm 1 : 160 bits) et MD5 (Message-Digest algorithm 5, 128 bits, plus ancien et moins sûr, voir figure 6) sont des fonctions de hachage utilisées fréquemment. Le SHA-2 (SHA-256, SHA-384 ou SHA-512 bits au choix) est d'ores et déjà prêt s'il faut abandonner aussi le SHA-1.

Application au contrôle d'accès

Un mot de passe ne doit pas être stocké en clair sur une machine pour des raisons de sécurité. Seul le résultat du hachage du mot de passe est donc stocké. Pour identifier un utilisateur, l'ordinateur compare l'empreinte du mot de passe d'origine (stocké) avec l'empreinte du mot de passe saisi par l'utilisateur. Toutefois, cette manière de faire n'est pas complètement satisfaisante. Si deux utilisateurs décident d'utiliser le même mot de passe alors le condensé obtenu sera identique. Cette faille est potentiellement utilisable par trois méthodes :

- attaque par dictionnaire ;
- attaque par force brute ;
- attaque par table arc-en-ciel.

Lors d'une attaque par dictionnaire, on pourrait raisonnablement déduire que le mot de passe choisi par les deux utilisateurs est relativement facile à mémoriser.

Pour contrer ce type d'attaque, on ajoute une composante aléatoire lors de la génération initiale de l'empreinte. Cette composante, aussi appelée « sel », est souvent stockée en clair. On peut simplement utiliser l'heure de l'attribution du mot de passe ou un compteur qui varie selon l'utilisateur. Le mot de passe est ensuite mélangé avec le sel, cette étape varie selon le système employé. Une

Tournez la page S.V.P.

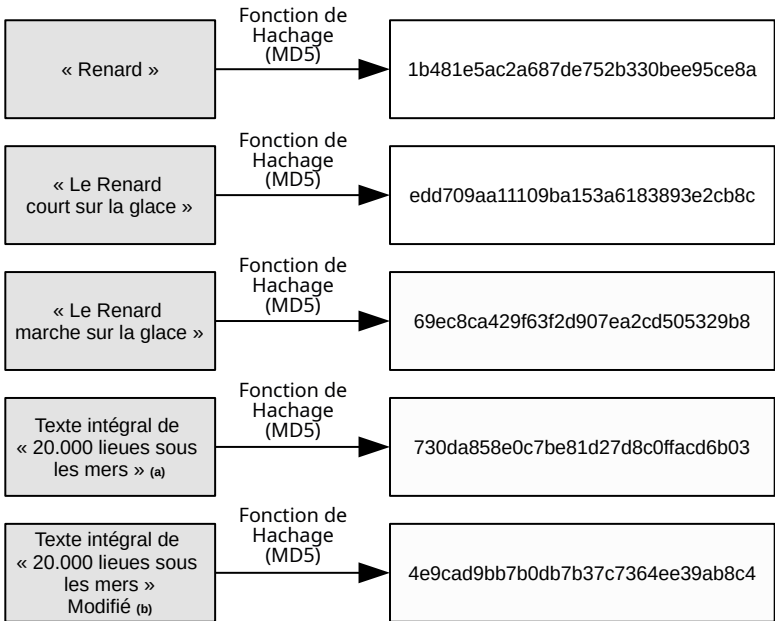


Figure 6: Exemples de hachages de textes par la fonction MD5; (a) le texte utilisé est la version libre de Vingt mille lieues sous les mers du projet Gutenberg2 ; (b) la version modifiée est le même fichier texte, le 10e caractère de la 1000e ligne ayant été remplacé par le caractère ∗.

méthode simple est de concaténer le mot de passe avec le sel. Le sel n'étant pas identique pour deux utilisateurs, on obtiendra deux signatures différentes avec le même mot de passe. Cela réduit fortement la marge d'une attaque via un dictionnaire.

DT4 : Salage des mots de passe

Extrait de https://fr.wikipedia.org/wiki/Salage_(cryptographie) consulté le 15/09/21 et légèrement adapté.

Objectif du salage

Le salage est une méthode permettant de renforcer la sécurité des informations qui sont destinées à être hachées (par exemple des mots de passe) en y ajoutant une donnée supplémentaire afin d'empêcher que deux informations identiques conduisent à la même empreinte (la résultante d'une fonction de hachage). Le but du salage est de lutter contre les attaques par analyse fréquentielle, les attaques utilisant des rainbow tables (tables arc-en-ciel), les attaques par dictionnaire et les attaques par force brute. Pour ces deux dernières attaques, le salage est efficace quand le sel (la valeur utilisée dans le salage) utilisé n'est pas connu de l'attaquant ou lorsque l'attaque vise un nombre important de données hachées toutes salées différemment.

Initialisation

Le salage consiste à concaténer le mot de passe avec le sel, une chaîne de caractères quelconque, le plus souvent aléatoire. Le salage peut être statique : chaque mot de passe est salé avec la même chaîne de caractères (mais ce type de salage est considéré comme dépassé), ou dynamique : chaque mot de passe est salé aléatoirement (cela empêchera à deux utilisateurs d'avoir la même empreinte s'ils ont le même mot de passe).

Dans le cas où le salage est dynamique, chaque enregistrement de la table de mots de passe du système d'authentification peut contenir les trois informations suivantes :

```
identifiant | hachage(sel + mot de passe) | sel
```

Exemple

Prenons par exemple le mot de passe « Wikipedia », qui utilisé avec l'algorithme de hachage SHA-256 produit (valeur en hexadécimal) :
`d38b38a2dd476e045c299e8ee5d6466834456d97bd592a71746b423a6a05f386`.

Utilisons un salage du mot de passe en y ajoutant le sel « S_8u ». En hachant « S_8uWikipedia », le hachage est maintenant :
`33bf32c0e27f6361d21287df83f19a23c5b2d0e7f25e1899ef8c3621640f8907`.

Le hash salé est très différent du hash non salé.

Utilisation

L'utilisateur entre son identifiant et son mot de passe brut. À partir de l'identifiant, le système d'authentification retrouve la valeur du sel associé, il concatène le sel et mot de passe brut, traite la chaîne de caractères obtenue par la fonction de hachage cryptographique puis compare le résultat avec le mot de passe salé et haché enregistré dans la table des mots de passe.

DT5 : Signature HMAC SHA-256

L'objectif de l'algorithme de signature HMAC SHA-256 est d'assurer l'authentification et l'intégrité d'une donnée. La méthode utilise une clé secrète, partagée entre les participants à la conversation. La donnée à signer est hachée, en utilisant la fonction de hachage SHA-256. Le condensé résultant est combiné, en utilisant l'algorithme HMAC avec la clé secrète. Le résultat donne une séquence binaire de la même taille que le condensé (32 octets ici). C'est ce résultat qui constitue la signature S du message M avec la clé K : $S = \text{HMAC}(\text{SHA256}(M), K)$.

Le message M est ensuite transmis, accompagné de sa signature S.

Le destinataire du message, accompagné de sa signature, possède lui aussi la clé secrète K. Il est en mesure de faire le calcul précédent, et de vérifier qu'il obtient bien la signature fournie. Les propriétés cryptographiques de la fonction de hachage assurent l'intégrité du message, et l'algorithme HMAC assure que l'expéditeur est bien en possession de la clé secrète (authentification).

Il est naturellement impossible en pratique de remonter au message depuis le condensé, ni à la clé depuis la signature et le message.

DT6 : The Base45 Data encoding

Source : https://datatracker.ietf.org/doc/rfc9285/ consultée le 15/08/2022

Authors : Patrik Faltstrom, Fredrik Ljunggren, Dirk-Willem van Gulik

A 45-character subset of US-ASCII is used; the 45 characters usable in a QR code in Alphanumeric mode [...].
Base45 encodes 2 bytes in 3 characters, compared to Base64, which encodes 3 bytes in 4 characters.

For encoding, two bytes $[a, b]$ MUST be interpreted as a number n in base 256, i.e. as an unsigned integer over 16 bits so that the number $n = (a \times 256) + b$.

This number n is converted to base 45 $[c, d, e]$ so that $n = c + (d \times 45) + (e \times 45 \times 45)$.
Note the order of c, d and e which are chosen so that the left-most $[c]$ is the least significant.

The values c, d, and e are then looked up [in the following table] to produce a three character string. The process is reversed when decoding.

For encoding a single byte $[a]$, it MUST be interpreted as a base 256 number, i.e. as an unsigned integer over 8 bits. That integer MUST be converted to base 45 $[cd]$ so that $a = c + (45 \times d)$. The values c and d are then looked up in [the following table] to produce a two-character string.

A byte string $[abcd...xyz]$ with arbitrary content and arbitrary length MUST be encoded as follows: From left to right pairs of bytes MUST be encoded as described above. If the number of bytes is even, then the encoded form is a string with a length that is evenly divisible by 3. If the number of bytes is odd, then the last (rightmost) byte MUST be encoded on two characters as described above.

For decoding a Base45 encoded string the inverse operations are performed.

When to Use and Not Use Base45

If binary data is to be stored in a QR code, the suggested mechanism is to use the Alphanumeric mode that uses 11 bits for 2 characters as defined in Section 7.3.4 of [ISO18004]. The Extended Channel Interpretation (ECI) mode indicator for this encoding is 0010.

On the other hand if the data is to be sent via some other transport, a transport encoding suitable for that transport should be used instead of Base45. For example, it is not recommended to first encode data in Base45 and then encode the resulting string in Base64 if the data is to be sent via email. Instead, the Base45 encoding should be removed, and the data itself should be encoded in Base64.

The Alphabet Used in Base45

The Alphanumeric mode is defined to use 45 characters as specified in this alphabet.

Table : The Base45 Alphabet

Value	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22
String	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J	K	L	M
Value	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	
String	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	␣	$	%	*	+	-	.	/	:	

Encoding examples

It should be noted that although the examples are all text, Base45 is an encoding for binary data where each octet can have any value 0-255.

Encoding example 1: The string `"AB"` is the byte sequence `[[65 66]]`. If we look at all 16 bits, we get $65 \times 256 + 66 = 16706$.
16706 equals $11 + (11 \times 45) + (8 \times 45 \times 45)$, so the sequence in base 45 is `[11 11 8]`. Referring to [the previous table], we get the encoded string `"BB8"`.

`AB`	Initial string
`[[65 66]]`	Decimal value
`[16706]`	Value in base 16
`[11 11 8]`	Value in base 45
`BB8`	Encoded string

[…]

Encoding example 3: The string "base-45" as ASCII is the byte sequence [[98 97] [115 101] [45 52] [53]]. If we look at this two bytes at a time, we get [25185 29541 11572 53]. Note the 53 for the last byte. When looking at the values in base 45, we get [[30 19 12] [21 26 14] [7 32 5] [8 1]] where the last byte is represented by two values. Referring to [the Base45 Alphabet Table], we get the encoded string "UJCLQE7W581".

base-45	Initial string
[[98 97] [115 101] [45 52] [53]]	Decimal value
[25185 29541 11572 53]	Value in base 16
[[30 19 12] [21 26 14] [7 32 5] [8 1]]	Value in base 45
UJCLQE7W581	Encoded string

Decoding examples

Decoding example 1: The string "QED8WEX0" represents, when looked up in [the Base45 Alphabet Table], the values [26 14 13 8 32 14 33 0]. We arrange the numbers in chunks of three, except for the last one which can be two numbers, and get [[26 14 13] [8 32 14] [33 0]]. In base 45, we get [26981 29798 33] where the bytes are [[105 101] [116 102] [33]]. If we look at the ASCII values, we get the string "ietf!".

DT7 : Trames NMEA 0183

Extrait et adapté de https://fr.wikipedia.org/wiki/NMEA_0183, consulté le 14/08/22

Les trames NMEA sont codées au format ASCII et sont de la forme :

```
$<talker ID><trame type>[,<data>,<data>]*<checksum>
```

Décodage :

Champs	Longueur (octets)	Signification
$	1	Marqueur de début de trame
<talker ID>	2	Équipement ayant généré la trame NMEA
<trame type>	3	Code identifiant le contenu de la trame
<data>	variable	Charge utile dont le contenu est défini par *Trame type*. Chaque valeur est séparée par le caractère ,
*	1	Séparateur de checksum
<checksum>	2	Somme de contrôle générée par un ou exclusif de tous les caractères situés entre $ et * (exclus)
Fin de ligne	2	Caractères *carriage return* + *line feed* marquant un retour à la ligne (<CR><LF> soit <0x0D><0x0A>)

La longueur totale d'une trame ne peut excéder 82 octets.

talker ID peut valoir BD, GB (Beidou), GA (Galileo), GP (GPS), GS (Glonnass), GN (GPS + Glonnass) selon le type de système GNSS utilisé.

Tournez la page S.V.P.

<trame type> peut prendre (entre autres) une des valeurs suivantes :

- GGA : horodatage, position et informations sur le fix
- GSA : mode opératoire, nombre de satellites utilisés, et valeurs de dilution de précision
- GSV : position (élévation, azimuth…) des satellites visibles

Dans le cas d'une trame GGA, on trouve 14 champs entre GGA et le * annonçant le checksum, séparés par des virgules :

```
$GPGGA,064036.289,4836.5375,N,00740.9373,E,1,04,3.2,200.2,M,,,,0000*0E
```

- 064036.289 : horaire d'émission de la trame, ici 06 h 40 min 36.298 s
- 4836.5375 : latitude ddmm.mmmm, ici 48° 36.5375'. Si la latitude vaut $x°y'$ (degrés, minutes), le nombre dans la trame vaudra $x \times 100 + y$
- N : latitude Nord (ou pourrait trouver un S)
- 00740.9373 : longitude dddmm.mmmm, ici 7° et 40.9373'
- E : longitude Est (ou pourrait trouver un W pour *West*)
- les 9 champs suivants n'ont pas d'importance ici
- * : séparateur de checksum
- 0E : Somme de contrôle de parité, un simple XOR sur les caractères situés strictement entre $ et *

Dans le cas d'une trame GSA, on trouve 17 champs entre GPA et le * annonçant le checksum :

```
$GPGSA,A,3,04,05,,09,12,,,24,,,,,2.5,1.3,2.1*39
```

- A : sélection Automatique 2D ou 3D du FIX (A=automatique, M=manuel)
- 3 : fix 3D
- 04,05... : numéros des satellites utilisés pour le FIX (12 champs car 12 satellites maximum)
- 2.5 : dilution de précision (PDOP)
- 1.3 : dilution de précision horizontale (HDOP)
- 2.1 : dilution de précision verticale (VDOP)
- * : séparateur de checksum
- 39 : somme de contrôle de parité, un simple XOR sur les caractères situés strictement entre $ et *

Dans le cas d'une trame GSV, on trouve 19 champs entre GSV et le * annonçant le checksum :

```
$GPGSV,2,1,08,01,40,083,46,02,17,308,41,12,07,344,39,14,22,228,45*75
```

- 2 : nombre de trames GSV pour obtenir les données complètes (1, 2 ou 3)
- 1 : ici, trame 1 sur les 2 trames attendues
- 08 : Nombre de satellites visibles en tout
- données sur le premier satellite :
  - 01 : N° d'identification du 1er satellite.
  - 40 : Élévation en degrés du 1er satellite.
  - 083 : Azimut en degrés du 1er satellite.
  - 46 : Force du signal du 1er satellite (Plus grand=meilleur)
- données sur le deuxième satellite : 02,17,308,41
- données sur le troisième satellite : 12,07,344,39
- données sur le quatrième satellite : 14,22,228,45
- * : séparateur de checksum
- 75 : somme de contrôle de parité, un simple XOR sur les caractères situés strictement entre $ et *

On a ainsi les données sur 4 satellites dans une trame. Le maximum de 3 trames correspond au maximum de 12 satellites.

DT8 : Table ASCII

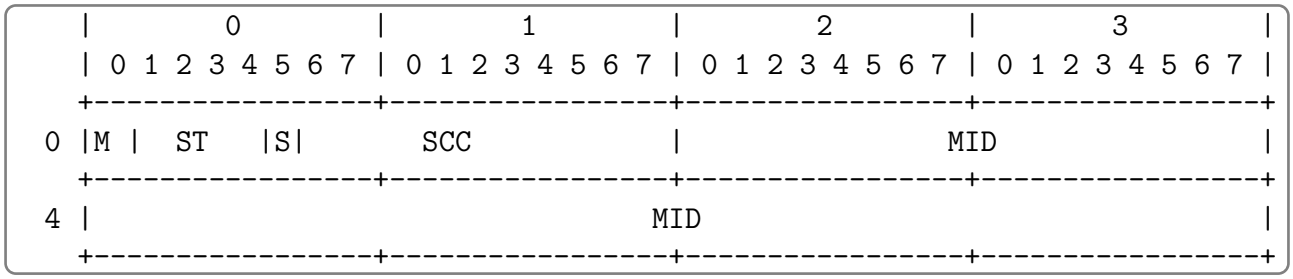
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2	␣	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	,	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Exemple : le code ASCII de `G`, en hexadécimal, s'écrit `47` (ligne `4`, colonne `7`).

DT9 : adresse *GeoNetworking*

Dans cette description, le bit 0 est le bit de poids fort.

La table suivante détaille la composition d'une adresse *GeoNetworking* (8 octets).



- `M` (1 bit) : 1 if address is manually configured
- `ST` (4 bits) : ITS Station type
  - bit 1
    - 0 : Vehicle ITS station
    - 1 : Roadside ITS station
  - bits 2,3,4
    - for roadside ITS station:
      - 0 : Traffic light
      - 1 : Ordinary roadside ITS station
    - for vehicle ITS station:
      - 0 : Bike
      - 1 : Motorbike
      - 2 : Car
      - 3 : Truck
      - 4 : Bus
- `S` (1 bit) : Public (0) or Private (1) ITS station
- `SCC` (10 bits) : ITS Country Code (*voir plus loin*)
- `MID` (48 bits) : address (LL_ADDR)

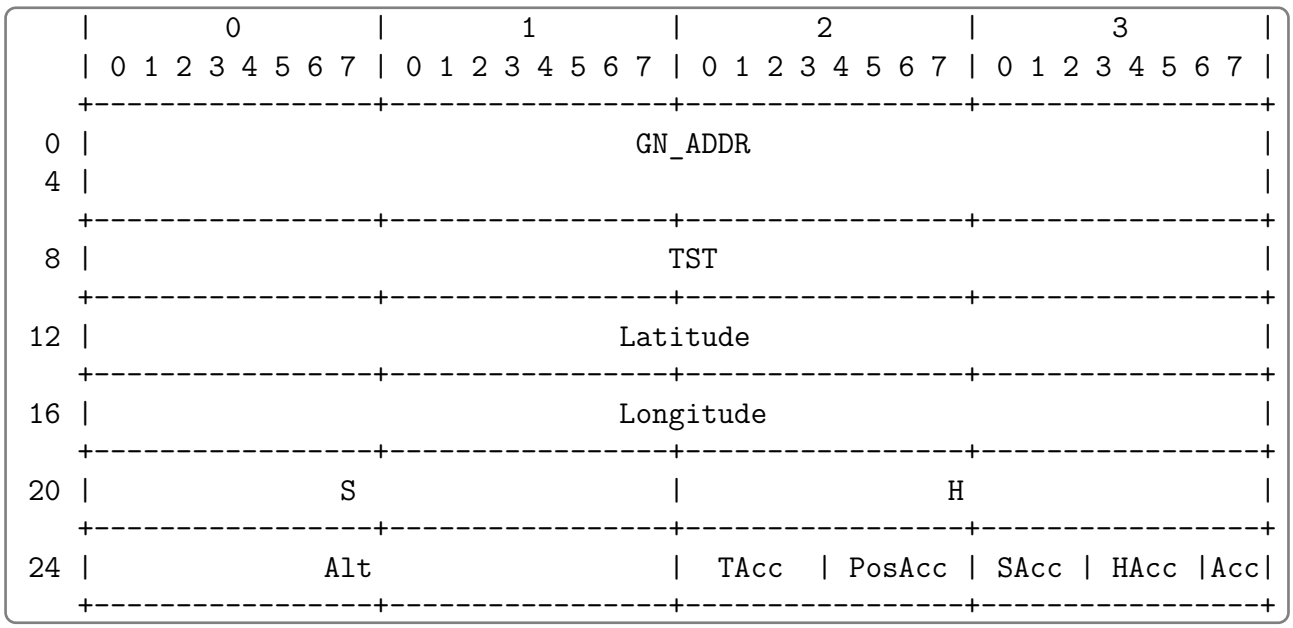
Tournez la page S.V.P.

Certaines valeurs possible pour le champs *ITS Country Code* sont données dans le tableau suivant :

Code	Pays	Code	Pays	Code	Pays
202	Grèce	234	Royaume-Uni	274	Islande
204	Pays-Bas	238	Danemark	283	Arménie
206	Belgique	240	Suède	284	Bulgarie
208	France	242	Norvège	286	Turquie
214	Espagne	244	Finlande	293	Slovénie
216	Hongrie	260	Pologne	295	Liechtenstein
222	Italie	268	Portugal	302	Canada
228	Suisse	270	Luxembourg	310	USA
232	Autriche	272	Irlande		

DT10 : Long position vector

Dans cette description, le bit 0 est le bit de poids fort.

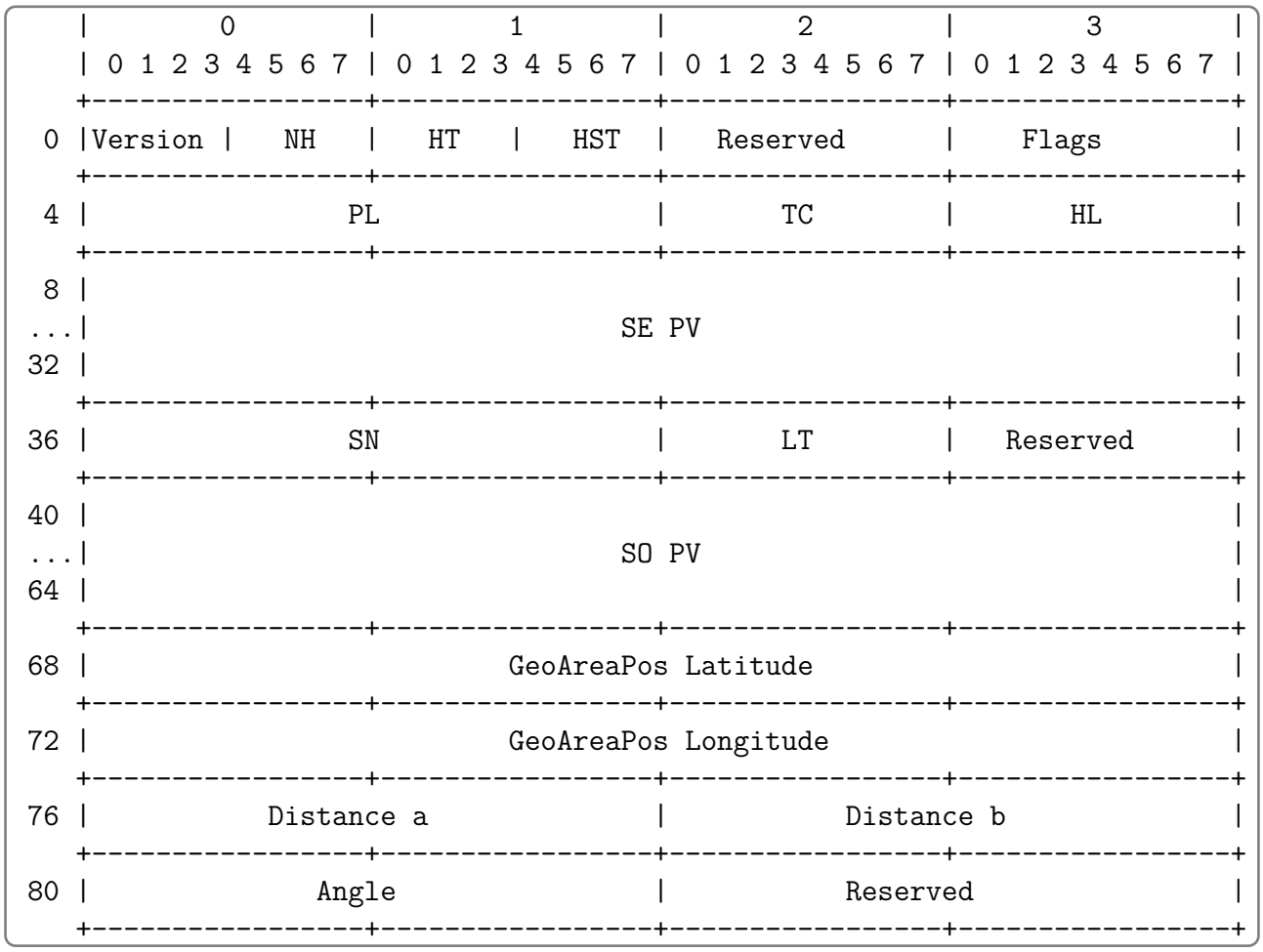


- `GN_ADDR` : Network address (8 bytes)
- `TST` : Express the time in milliseconds at which position of the ITS station was acquired. Time is encoded as : `TST = TST(UET) mod 2^32` where `TST(UET)` is the number of milliseconds since `1970-01-01T00:00`
- `Latitude` : Latitude expressed in 1/10 micro degree
- `Longitude` : Longitude expressed in 1/10 micro degree
- `S` : Speed expressed in units of 0.1 meters per second
- `H` : Heading expressed in 0.1 degrees from North
- `Alt` : Altitude (meters)
- `TAcc` : Accuracy indicator (4 bits)
- `PosAcc` : Accuracy indicator (4 bits)
- `SAcc` : Speed accuracy indicator (3 bits)
- `HAcc` : Heading accuracy indicator (3 bits)
- `Acc` : Altitude accuracy indicator (2 bits)

DT11 : Entête étendu GeoNetworking

Dans cette description, le bit 0 est le bit de poids fort.

Le contenu détaillé est l'entête d'un paquet de type `GeoBroadcast`.



- `Version` (4 bits) : version of the GeoNetworking Protocol
- `NH` (4 bits) : type of header following the GeoNetworking header
- `HT` (4 bits) : type of GeoAdhoc header (4 for `GeoBroadcast`)
- `HST` (4 bits) : subtype of GeoAdhoc header (0 for a Circular Area)
- `Flags` : Type of ITS station (bit 6), other bits set to zero
- `PL` : Length of the network header payload
- `TC` : Traffic class
- `HL` : Time to live decremented by 1 each router
- `SE PV` : Long position vector of the sender
- `SN` : Sequence Number
- `LT` : Lifetime field : maximum tolerable time a packet can be buffered until it reaches its destination
- `SO PV` : Long position vector of the source
- `GeoAreaPos Latitude` : WGS-84 latitude of the area in 1/10 micro degree
- `GeoAreaPos Longitude` : WGS-84 longitude of the area in 1/10 micro degree
- `Distance a` : distance a of the geometric shape
- `Distance b` : distance b of the geometric shape
- `Angle` : angle of the geometric shape

Si la zone ciblée est un cercle (`HST` à 0), `Distance a` vaut le rayon du cercle, `Distance b` et `Angle`

Tournez la page S.V.P.

sont à zéro et la latitude et la longitude données sont le centre du cercle.

DT12 : Simple GeoBroadcast forwarding algorithm with line forwarding

Extrait de *ETSI TS 102 636-4-1 V1.1.1 (2011-06)*

The algorithm utilizes the function $F(x, y)$ to determine whether the GeoAdhoc router is located inside, at the border or outside the geographical target area carried in the GeoBroadcast packet header. If the GeoAdhoc router is inside or at the border of the area, the packet shall be re-broadcasted. If it is outside the area, the packet shall be forwarded by the Greedy Forwarding algorithm (GF Algorithm).

With the Greedy Forwarding (GF) algorithm, the GeoAdhoc router uses the location information of the destination carried in the GeoUnicast packet header and selects one of the neighbours as the next hop. The algorithm applies the most forward within radius (MFR) policy, which selects the neighbour with the smallest geographical distance to the destination, thus providing the greatest progress when the GeoUnicast packet is forwarded.

If no neighbour with greater progress than the local GeoAdhoc router exists, the packet has reached a local optimum.

```
SIMPLE GEOBROADCAST FORWARDING ALGORITHM
========================================

-- P is the GeoNetworking packet to be forwarded
-- LAT and LONG are latitude and longitude of the LPV, respectively
-- DAp is the destination area in the GeoNetworking packet to be forwarded
-- A is the centre point of the destination area DAp
-- LL_ADDR is the link layer address that identifies the next hop
-- of the GeoNetworking packet
-- BCAST is the broadcast LL address
-- GREEDY() is the GF algorithm
LL_ADDR = 0
Calculate F(LAT, LONG)
IF (F≥0) THEN
    RETURN LL_ADDR = BCAST
ELSE
    RETURN LL_ADDR = GREEDY(A) or 0
ENDIF
```

`GREEDY` (`GF ALGORITHM`) est détaillé page suivante.

```
GF ALGORITHM
==============
-- P is the GeoUnicast packet to be forwarded
-- i is the i-th LocTE
-- NH is the LocTE identified as next hop
-- NH_LL_ADDR is the link layer address of the next hop
-- LPV is the local position vector
-- PVp is the destination position vector in the GeoNetworking
--     packet to be forwarded
-- PVi is the position vector of the i-th LocTE
MFR = DIST(PVp, LPV)
FOR (i ∈ LocT)
    IF (i.IS_NEIGHBOUR) THEN
        IF (DIST(PVp, PVi) < MFR) THEN
            NH ← i
            MFR ← DIST(PVp, PVi)
        ENDIF
    ENDIF
ENDFOR
IF (MFR < DIST(PVp, LPV)) THEN
    SET NH_LL_ADDR = NH.LL_ADDR
ELSEIF
    LOCAL OPTIMUM
    SET NH_LL_ADDR = 0
ENDIF
```

Modèle CMEN-DOC v2 ©NEOPTEC

Nom de famille :
(Suivi, s'il y a lieu, du nom d'usage)

Prénom(s) :

Numéro Inscription :

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

Né(e) le : ☐☐ / ☐☐ / ☐☐☐☐

(Remplir cette partie à l'aide de la notice)

Concours / Examen : …….. **Section/Spécialité/Série :** ……

Epreuve : …….. **Matière :** …….. **Session :** ……..

CONSIGNES

- *Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.*
- *Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.*
- *Numéroter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre.*
- *Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.*
- *N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.*

DR1 à DR5

Tous les documents réponses sont à rendre, même non complétés.

NE RIEN ECRIRE DANS CE CADRE

Document réponse DR1

```
# dans outils.py
def hash_with_salt(salt: str, password: str) -> str:
    """
    >>> hash_with_salt("S_8u", "Wikipedia")
    'S_8u:33bf32c0e27f6361d21287df83f19a23c5b2d0e7f25e1899ef8c3621640f8907'
    """
    b_password = password.encode("utf8")
    b_salt = ........................
    hashed_password = hashlib.sha256(b_salt + ..........).hexdigest()
    return salt + ":" + ......................

# dans database.py
class Database:

    def store_password(self, client_email: str, password: str) -> bool:
        salt = outils.random_chars(16)
        return self.update_pwd(client_email, ............................)
```

Document réponse DR2

```
class Database:

    def check_password(self, client_email: str, password: str) -> bool:
        try:
            stored_hashed_password = ............................
        except DatabaseValueError:
            return False
        ....... = stored_hashed_password.split(":")[0]
        return ......................... == stored_hashed_password
```

Document réponse DR3

```
class CourseClient:

    @classmethod
    def from_bytes(cls, b: bytes) -> "CourseClient":
        """
        Méthode de classe qui renvoie un nouvel objet `CourseClient`
        à partir de sa représentation compacte
        """
        course, client = struct.unpack(">ii", b[:8])
        pos_depart = Position.from_bytes(.....................)
        h_depart = ......................
        pos_arrivee = ...................
        h_arrivee = ......................
        heure_depart = datetime.datetime(*h_depart)
        heure_arrivee = datetime.datetime(*h_arrivee)
        return cls(.............................)
```

Document réponse DR4

```
def check_sign(data_sig: bytes) -> bool:
    """
    Vérifie que les données sont correctement signées
    (ont été correctement produites par `sign`)
    """
    sig_type = data_sig[0]
    if sig_type == 1:
        ....................................................
        ....................................................
        ....................................................
        ....................................................
        ....................................................
        ....................................................
        ....................................................
    else:
        return False
```


Tournez la page S.V.P.

Document réponse DR5

```
def to_b45(a: int, b: Optional[int]=None)  -> Union[Tuple[int, int],
                                               Tuple[int, int, int]]:
    if b is None:
        n = a
        c, d = n % 45, n // 45
        return (c, d)
    else:
        ..........................................................
        ..........................................................
        ..........................................................
        ..........................................................
        return (c, d, e)

def b45_to_str(b45lst: list) -> str:
    chars = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:"
    return "".join(chars[k] for k in b45lst)

def base45_encode(data: bytes) -> str:
    """
    Encode une séquence d'octets en chaîne Base45
    """
    lst_res = []
    lst_bytes = list(data)
    if len(data) % 2 == ...........:
        lst_bytes.append(None)
    for a, b in zip(lst_bytes[0::2], lst_bytes[1::2]):
        ..........................................................
        ..........................................................
    return "".join(lst_res)
```

Modèle CMEN-DOC v2 ©NEOPTEC

Nom de famille :
(Suivi, s'il y a lieu, du nom d'usage)

Prénom(s) :

Numéro Inscription :

Né(e) le : ☐☐/☐☐/☐☐☐☐

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

(Remplir cette partie à l'aide de la notice)

Concours / Examen : …….. **Section/Spécialité/Série :** ……

Epreuve : …….. **Matière :** …….. **Session :** ……..

CONSIGNES

- *Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.*
- *Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.*
- *Numéroter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre.*
- *Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.*
- *N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.*

DR6 - DR7

Tous les documents réponses sont à rendre, même non complétés.

NE RIEN ECRIRE DANS CE CADRE

Document réponse DR6

```
typedef struct {
  float latitude, longitude;
  int heures, minutes;
  float secondes;
} position;

float read_deg(char str[]) {
    float tmp;
    int deg;
    float minutes;
    sscanf(str, "%f", &tmp);
    deg = tmp / 100;
    minutes = tmp - deg * 100;
    return deg + minutes / 60;
}

void read_latitude(char str[], float * latitude) {
    *latitude = read_deg(str);
    if (str[10] == 'S') *latitude = -*latitude;
}

void read_longitude(char str[], float * longitude) {
    ..........................................................
    ..........................................................
    ..........................................................
    ..........................................................
    ..........................................................
    ..........................................................
}
void read_timestamp(char str[], int * h, int * m, float *s) {
    ..........................................................
    ..........................................................
    ..........................................................
    ..........................................................
    ..........................................................
    ..........................................................
}
```

Le document réponse 6 continue page suivante

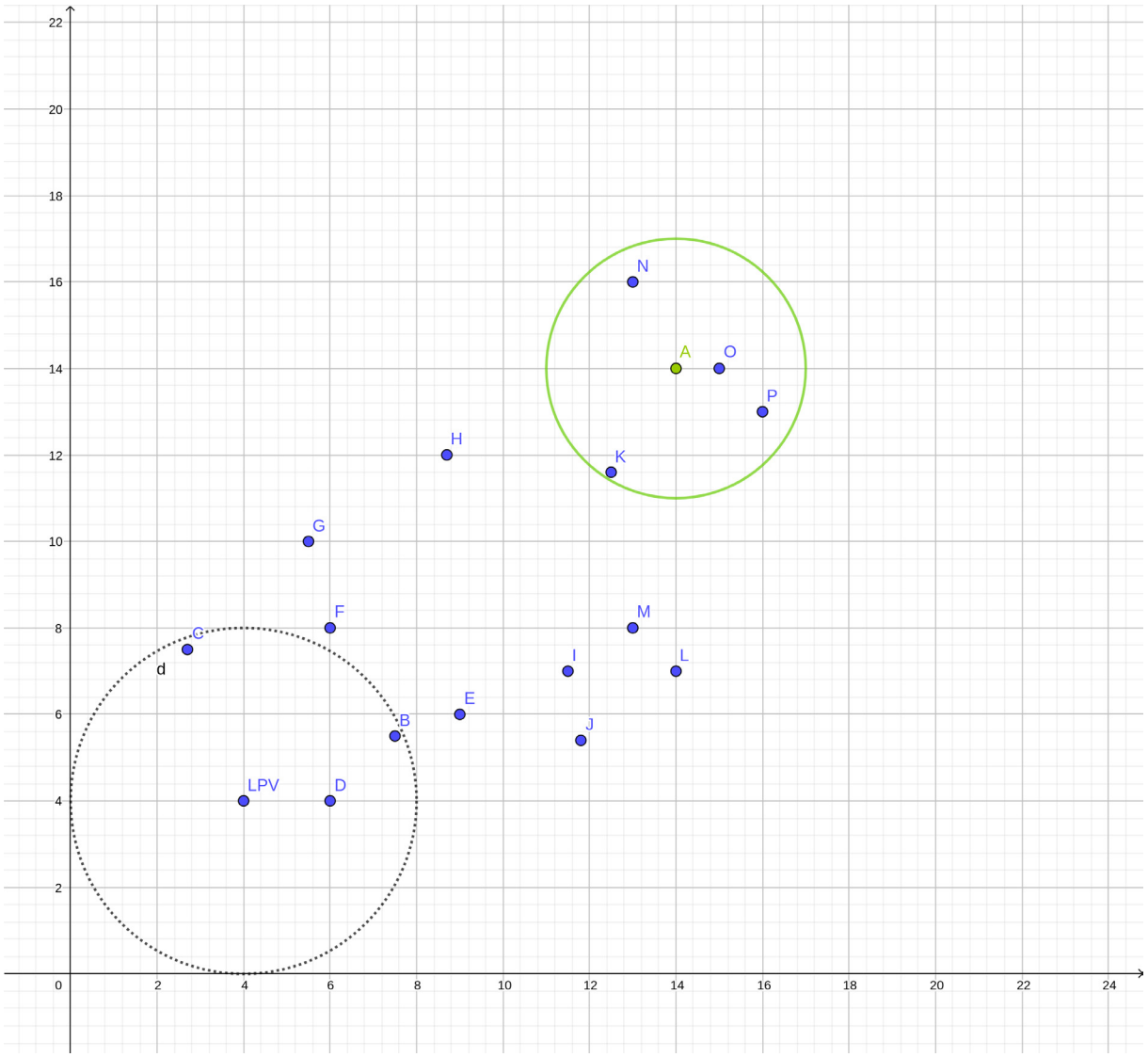
```
int check_crc(char str[]) {
    ....................................................
    ....................................................
    ....................................................
    ....................................................
    ....................................................
    ....................................................
}

/* Fonction principale, prenant en paramètre une trame NMEA
   complète et un pointeur vers une structure de position.
   Cette structure est remplie par la fonction, qui renvoie 1
   si un décodage a été effectué, et 0 sinon (par exemple si
   ce n'est pas une trame du bon type)
*/
int get_position(char nmea_frame[], position * pos) {
    if (nmea_frame[0] != '$') return 0;
    if (!check_crc(nmea_frame)) return 0;
    ....................................................
    ....................................................
    ....................................................
    ....................................................
    ....................................................
    ....................................................
}
```

Document réponse DR7

	A	B	C	D	E	F	G	H
A		10.7	13.0	12.8	9.4	10.0	9.4	5.7
B	10.7		5.2	2.1	1.6	2.9	4.9	6.6
C	13.0	5.2		4.8	6.5	3.3	3.8	7.5
D	12.8	2.1	4.8		3.6	4.0	6.0	8.4
E	9.4	1.6	6.5	3.6		3.6	5.3	6.0
F	10.0	2.9	3.3	4.0	3.6		2.1	4.8
G	9.4	4.9	3.8	6.0	5.3	2.1		3.8
H	5.7	6.6	7.5	8.4	6.0	4.8	3.8	
I	7.4	4.3	8.8	6.3	2.7	5.6	6.7	5.7
J	8.9	4.3	9.3	6.0	2.9	6.4	7.8	7.3
K	2.8	7.9	10.6	10.0	6.6	7.4	7.2	3.8
L	7.0	6.7	11.3	8.5	5.1	8.1	9.0	7.3
M	6.1	6.0	10.3	8.1	4.5	7.0	7.8	5.9
N	2.2	11.9	13.4	13.9	10.8	10.6	9.6	5.9
O	1.0	11.3	13.9	13.5	10.0	10.8	10.3	6.6
P	2.2	11.3	14.4	13.5	9.9	11.2	10.9	7.4
LPV	14.1	3.8	3.7	2.0	5.4	4.5	6.2	9.3

Tournez la page S.V.P.

	I	J	K	L	M	N	O	P	LPV
A	7.4	8.9	2.8	7.0	6.1	2.2	1.0	2.2	14.1
B	4.3	4.3	7.9	6.7	6.0	11.9	11.3	11.3	3.8
C	8.8	9.3	10.6	11.3	10.3	13.4	13.9	14.4	3.7
D	6.3	6.0	10.0	8.5	8.1	13.9	13.5	13.5	2.0
E	2.7	2.9	6.6	5.1	4.5	10.8	10.0	9.9	5.4
F	5.6	6.4	7.4	8.1	7.0	10.6	10.8	11.2	4.5
G	6.7	7.8	7.2	9.0	7.8	9.6	10.3	10.9	6.2
H	5.7	7.3	3.8	7.3	5.9	5.9	6.6	7.4	9.3
I		1.6	4.7	2.5	1.8	9.1	7.8	7.5	8.1
J	1.6		6.2	2.7	2.9	10.7	9.2	8.7	7.9
K	4.7	6.2		4.8	3.6	4.4	3.5	3.8	11.4
L	2.5	2.7	4.8		1.4	9.1	7.1	6.3	10.4
M	1.8	2.9	3.6	1.4		8.0	6.3	5.8	9.8
N	9.1	10.7	4.4	9.1	8.0		2.8	4.2	15.0
O	7.8	9.2	3.5	7.1	6.3	2.8		1.4	14.9
P	7.5	8.7	3.8	6.3	5.8	4.2	1.4		15.0
LPV	8.1	7.9	11.4	10.4	9.8	15.0	14.9	15.0	



Le premier cercle est déjà tracé (en pointillés).