Application du protocole SVC pour la communication dans un réseau de transport public

Préface

Dans le cadre d'une collaboration du Groupe de Recherche ERISCS, on conduit un projet pour la RTM concernant la conception d'un système de communication sécurisé entre les bus et le(s) serveur(s) de surveillance.

Ce document détaille la conception et la réalisation d'une version améliorée par nos soins d'un protocole de communication s'appelant SVC [1], qui s'adapte le mieux au besoin de notre partenaire industriel.

Je me permets de remercier Messieurs T. Muntean, A. Février, l'équipe RTM et tous mes collègues de m'avoir aidé tout au long de ce travail.

Marseille, le 04/09/2016

Xuan Thong DANG

Table de contenu

I. Problématique

La Régie des Transports Marseillais (RTM) est la société qui est en charge du service de transports (bus, métro, etc) à Marseille – une de trois plus grandes villes de France. Dans les efforts d'évolution et d'amélioration de ses services, en particulier le réseau de transport en bus, la RTM a besoin d'y mettre en place un système de monitoring et surveillance temps-réel, qui est une des exigences de la spécification EBSF [2].

Pour supporter la communication entre les bus et les serveurs de la RTM, des réseaux sans-fil hétérogènes seront utilisés (wifi, 3G, 4G), sur lesquels doit être conçu un nouveau protocole de communication sécurisé. Des tels protocoles existent déjà sur le marché, mais rien n'a été conçu pour les besoins spécifiques rencontrés dans les réseaux de transport urbains.

Notre mission est donc d'analyser la nature de la communication, les avantages et désavantages des solutions existantes et de proposer une solution la plus adaptée aux spécifications des besoins.

II. Analyse

1. La nature de la communication entre les bus et le serveur

La communication dans le réseau de transport est un type de communication spécifique. Elle subit des contraintes et obligations imposées par la nature du service. Nous pouvons résumer ici les contraintes suivantes :

- **Basculement rapide de connexion** : en raison de la mobilité des bus, la connexion vers le serveur est souvent interrompue quand le bus sort d'une zone de couverture d'une antenne vers celle d'une autre, ou est empêchée par de grands obstacles physiques (bâtiments, tunnels, etc). Le changement du type de réseaux peut aussi être nécessaire dans le cas où le support du réseau courant ne peut pas desservir toutes les stations qui sont autour d'un point d'accès, ou les débits s'avèrent insuffisants pour répondre aux contraintes dynamique de communication des applications embarquées dans le bus (e.g. vidéo-surveillance).

- **Gros volume de données** : les données récoltées pendant les trajets du bus, concernant le statut de tous les équipements, la position cartographique du véhicule et les points d'accès, et notamment les images de vidéo-surveillance, doivent être envoyées vers le serveur avec des contraintes de temps spécifiques. Non seulement un réseau haut débit mais aussi un bon protocole de transmission qui peut réduire, dynamiquement si nécessaire, la surcharge de bande passante est requis.

- **Diversité de types et de priorités de données :** les données remontées vers le serveur viennent de sources différentes, certaines sont capables de tolérer des erreurs (par exemple quelques trames de vidéo-surveillance), d'autres requièrent strictement d'être livrées sans erreurs. On utilise aussi des priorités d'envoi, avec lesquelles les données sont traitées et envoyées dans le cas de congestion.

- **Temps réel :** toutes les données de surveillance doivent être transmises au serveur dès qu'elles sont disponibles, pour que les contrôleurs aient des réactions à temps en cas d'incidents. Aujourd'hui ces données sont enregistrées pendant les trajets et ne sont exploitables qu'après lecture à l'arrivée du bus au dépôt, souvent en fin de journée !

- **Sécurité :** les informations collectées dans le bus, y compris les images des passagers sont liées à la vie privée et doivent donc être transmises et exploitées de façon sécurisée. Une méthode de chiffrement appropriée est indispensable dans l'ensemble de la solution proposée et de bout-en-bout pour tout échange avec le serveur quelques soient les réseaux sans fil utilisés.

2. Les exigences d'un protocole de communication spécifique

Suite aux caractéristiques de la communication citées au-dessus, un protocole nouveau doit être spécifié qui s'adapte à ces besoins et doit posséder les propriétés suivantes:

- Sans connexion : une communication avec un basculement dynamique et efficace de connexion ne devrait pas utiliser un protocole de transport en mode connecté comme TCP, ce qui augmenterait la latence de connexion à cause de sa phase de négociation et des contrôles inutiles. Ainsi, un protocole de type UDP restera notre choix, mais un protocole « hybride » a été considéré comme remplaçant avec des contrôles simplifiés pour la garantie de livraison des messages.
- Reprise de connexion rapide : dans le même but de baisser les temps de connexion/reconnexion, on utilisera un couple d'identité pour distinguer toutes les connexions vers/depuis des hôtes. Une identité de session (sessionID) nous permettra d'identifier l'hôte, alors qu'une identité d'extrémité déterminera le bout de communication (une instance de l'application). À tout moment, ce couple d'informations nous permet de savoir à quelle application appartient chaque paquet échangé.
- Confidentialité persistante parfaite : pour bien protéger les identités des clients et les données des services, le protocole sécurisé doit supporter la **confidentialité persistante parfaite** (« perfect forward secrecy » en anglais). Comme ça, il garantit que la découverte par un adversaire de la clé privée d'un correspondant (secret à long terme) ne compromet pas la confidentialité des communications antérieures.
- Cryptographie elliptique : une courbe elliptique sera utilisée pour réduire la taille des clés en gardant le même niveau de sécurité, et économiser la bande passante.
- Indépendance de l'environnement support : le protocole est destiné à utiliser plusieurs plateformes et appareils, il faudrait donc être indépendant des versions du système d'exploitation et des bibliothèques externes, et fournir sa propre implémentation des algorithmes et services.
- Sécurité prouvée de bout-en-bout : les réseaux utilisés sont en général hétérogènes pour un réseau complexe de transport métropolitain ; les propriétés de sécurité exigées par les applications doivent être respectées de bout-en-bout pour le protocole de communication

3. Solutions existants et motivations pour un nouveau protocole

Les protocoles sécurisés « classiques » utilisés de nos jours sont basés essentiellement sur TCP : TLS, SSH, OpenVPN, etc…

Déjà, le TCP n'est pas un choix idéal pour notre solution. En effet, il ne satisfait pas la première caractéristique de la connexion car l'établissement et la reprise de la connexion TCP prend toujours du temps considérable. Le changement dynamique de support de connexion réseau es donc compromis. De plus, la garantie de livraison de paquets n'est pas nécessaire en permanence car souvent les données applicatives sont tolérantes aux pertes. Le compromis entre fiabilité et efficacité pour TCP augmente la latence de connexion, la rendant moins efficace à utiliser dans un contexte temps-réel.

On peut par contre trouver quelques versions UDP des protocoles ci-dessus (comme par exemple DTLS), ou ces protocoles offrent une option UDP eux-mêmes. Pour garder la même opérabilité qu'en TCP, des modifications supplémentaires sont introduites. Dans l'optique de minimiser la complexité, ce n'est pas non plus une bonne décision.

D'ailleurs, comme indiqué dans la définition initiale du protocole SVC par le groupe ERISCS[1], les protocoles mentionnés imposent encore des limites pour la protection d'identité de clients ou l'exposition des informations sensibles.

Dans cette optique est née l'idée de SVC, un nouveau protocole sécurisé qui surmonte ces limites en proposant une négociation simple et un découplage entre l'authentification et le chiffrement. SVC implémente les algorithmes de chiffrement les plus récents sans utiliser des sources externes, cela facilitera le processus d'administration et maintenance.

Pourtant, la version originale du SVC a besoin de quelques améliorations pour mieux s'adapter au problème posé par le contexte des applications des réseaux mobiles hétérogènes utilisés par la RTM dans les applications de transport métropolitain. Dans la section suivante, on propose une version modifiée du SVC et la conception de l'ensemble de la solution.

[1] G. Risterucci, T. Muntean, L. Mugwaneza. *A new Secure Virtual Connector approach for communication within large distributed systems.* ERISCS Research Group.

4. Les désavantages de la version originale du SVC

On a trouvé beaucoup d'idées innovantes dans la conception du SVC, parmi lesquelles on a choisi à développer l'idée de découpler l'authentification et l'autorisation, et l'idée de protection d'identité du client. Et pourtant, SVC présente dans la conception initiale des points faibles à améliorer. Dans la version originale proposée par ses auteurs, le protocole commence par une requête depuis le serveur vers le client (après un succès de connexion). Cette approche est justifiée comme une façon de réduire la durée de la négociation. Nous, on propose d'optimiser encore plus la phase d'initialisation de connexion car SVC consomme 4 exchanges de messages.

Le problème le plus pénalisant c'est le fait d'avoir l'étape d'échange de clés initiée par le serveur ce qui rend le protocole sensible aux attaques par rejeu. Une façon pour l'éviter, c'est de rajouter des timestamps, mais qui ne sera pas une bonne idée.

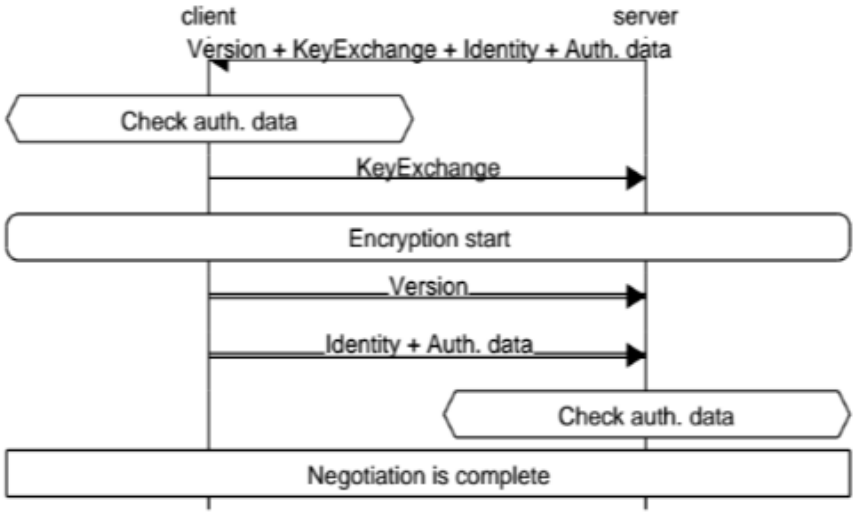

Figure 3.1 – Echange de messages dans la version originale du SVC

III. Solution

1. Une version modifiée du SVC

Avec seulement 3 échanges, on propose une nouvelle approche basée sur le principe du « compactage en 3 phases » qui garde encore les idées basiques du SVC.

Dans le diagramme suivant, on distingue les données « entre crochets », de celles qui sont en dehors. Les dernières sont des données de la négociation DH-STS, qui assurent l'autorisation du service, deviennent optionnelles en raison du découplage de service (à voir dans la section suivante). Les premières sont des données d'authentification dépendantes de l'application qui utilise SVC.

Dans la vérification de l'authenticité, l'approche « challenge – proof » est introduite pour éviter tout type d'attaque par rejeu. Le DH-STS est forcément imperméable [3], seulement vulnérable à quelques attaques du type « unknown key-share » [4].

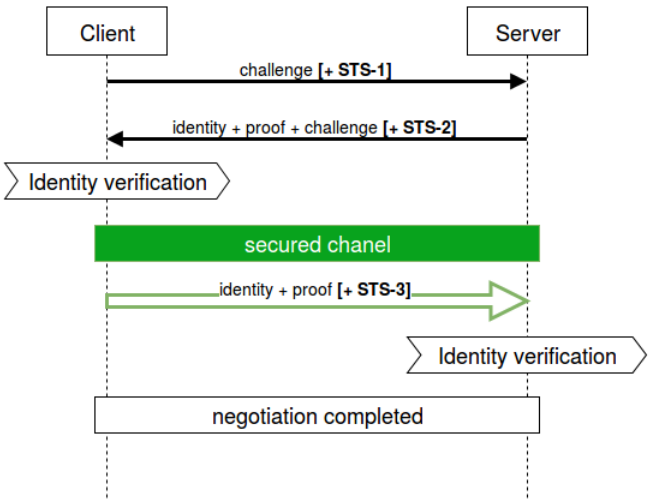

Figure 3.2 – Echange de messages dans la version modifiée du SVC

Comparée avec la version originale, la nouvelle que nous proposons ici montre des avantages supplémentaires:

- Moins d'échanges (3 contre 4)
- Résistant aux attaques par rejeu
- Découplage de service

2. L'architecture « service-applications »

À la naissance, SVC visait à servir une communication interne à une application, c'est-à-dire une communication entre les instances d'une même application. Dans un environnement multitâche, cette conception montre des limites:

- chaque connexion doit gérer ses propres (avec une même politique) paramètres de sécurité, chiffrement. Si on a plusieurs connexions vers un même hôte, toutes les étapes de négociation se refont.
- la mise à jour des protocoles de chiffrement du SVC impose la recompilation de toutes les applications qui l'utilisent.

L'idée est de découper la couche « application », qui est en charge de l'authentification et l'échange de données, avec une seule instance de la couche « daemon », qui s'occupe des services de chiffrement et de négociation. Comme ça, on remédie tout de suite aux problèmes posés :

- Les services gèrent toutes les connexions depuis et vers des hôtes. Quand une nouvelle connexion vers un même hôte est détectée, on utilise le service correspondant pour profiter d'un canal déjà sécurisé, laisse passer les échanges de clés.
- La mise à jour se fait simplement en replaçant et redémarrant l'instance du daemon. Toutes les applications restent intouchées.

Le diagramme ci-dessous présente l'architecture de la solution :

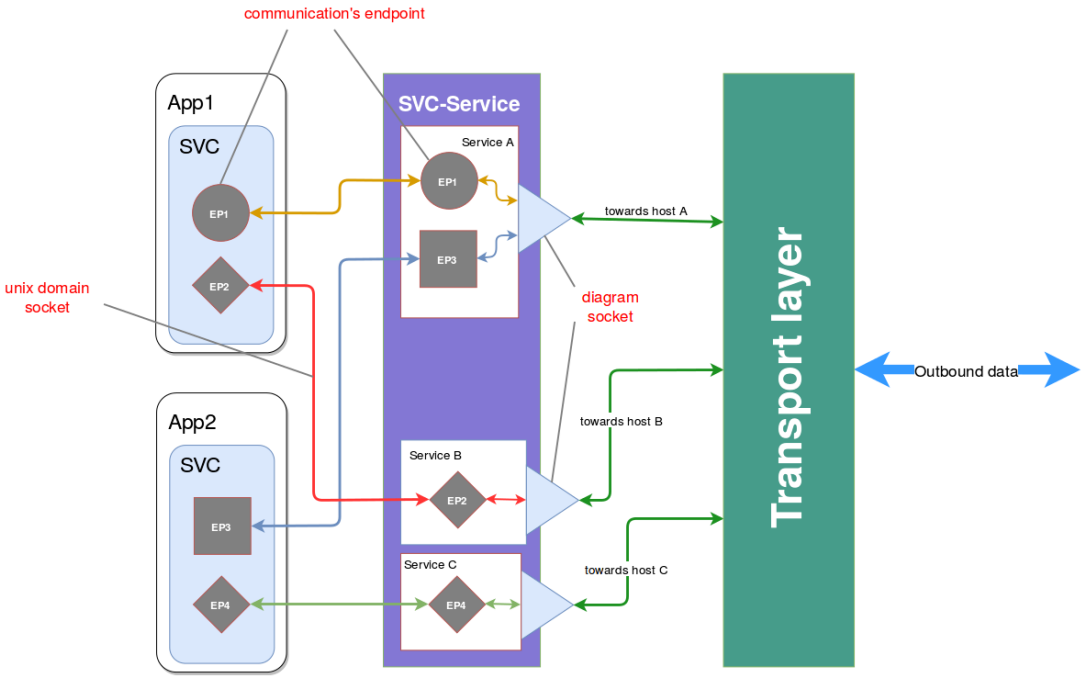

Figure 3.3 – L'architecture « service-applications » du SVC

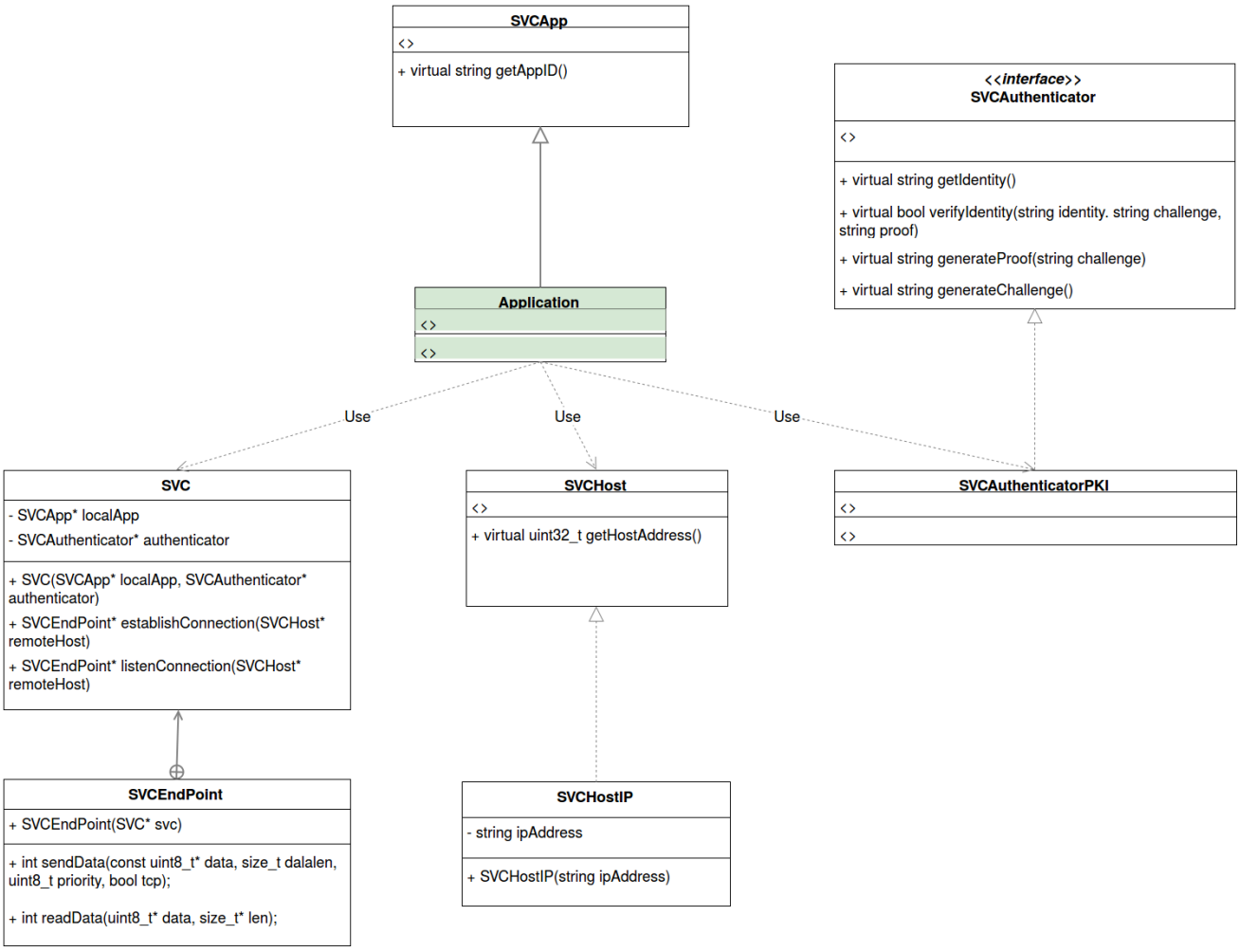

Figure 3.4 – Le diagramme de classe côté application

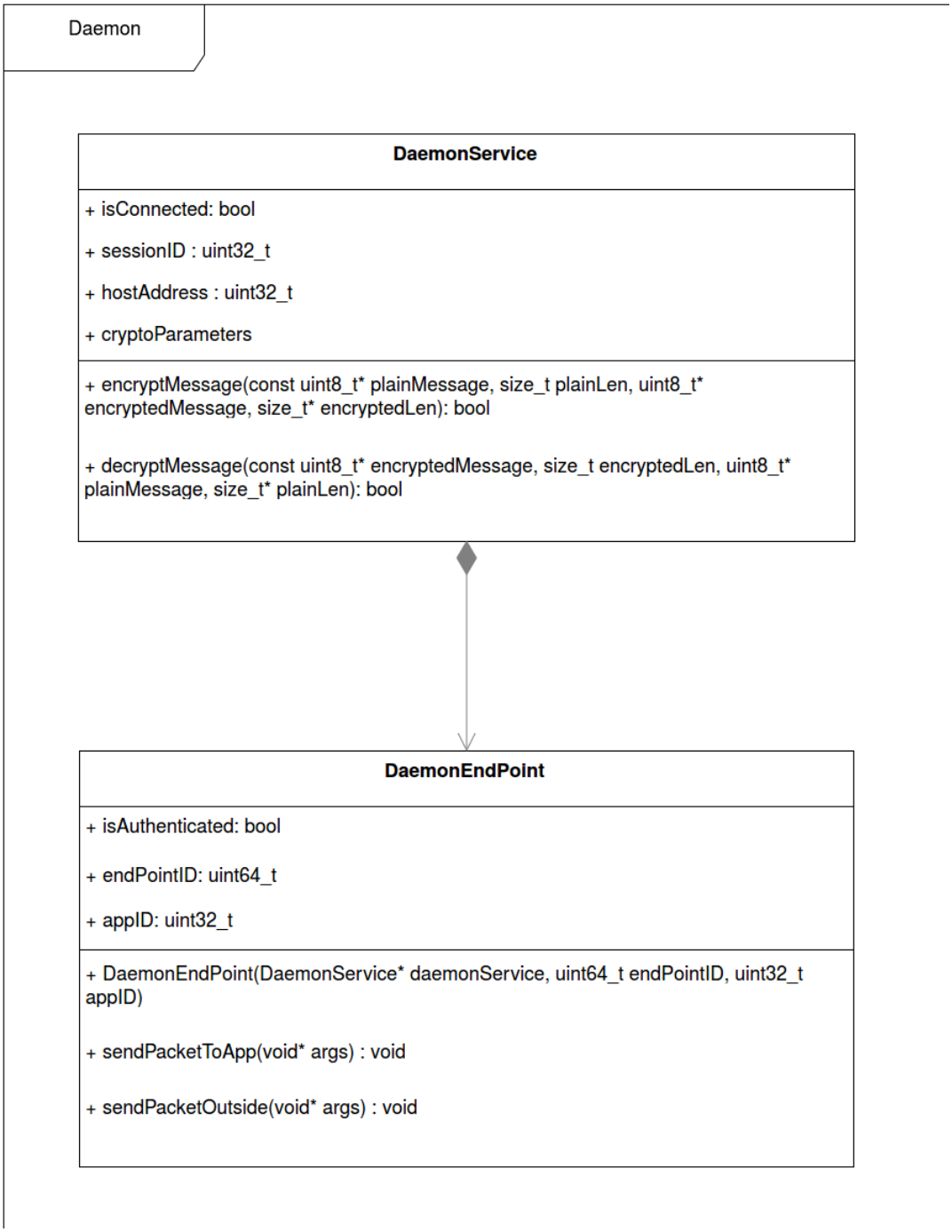

Figure 3.5 – Le diagramme classe côté service

3. La structure des trames SVC

Une trame SVC est encapsulée dans la partie de données d'une trame UDP. Il y a deux types de trames SVC : les trames de commandes servent à établir et gérer la connexion, les trames de données quant elles contiennent les données chiffrées des applications. Une trame du type « données » est toujours chiffrée. Le format de données dans ce type de trame est géré par l'application.

Quel que soit le type de trame, toutes les trames SVC commencent par :

- **Identité de session** – 4 octets: identité unique de la session. Une session est une connexion établie entre deux services (ou bien deux hôtes), qui partage les mêmes paramètres de chiffrement avec tous les chemins de connexion.

- **Identité d'extrémité** – 8 octets : identité d'une extrémité de connexion. Un service SVC utilise cette identité pour distinguer les paquets venant d'une même hôte. Un chemin de connexion est ce qui connecte 2 extrémités de communication, et ces extrémités partagent une identité unique.

L'utilisation de l'identité de session et d'extrémité permet la reprise rapide de connexion en cas de reconnexion.

Après l'identité de session et d'extrémité, succède un octet qui détermine le type de la trame et contient des paramètres pour la gestion de données.

- Données/commande: 7th bit, égale 1 si la trame est du type commande, 0 si la trame contient des données
- Réponse depuis daemon: 6th bit, égale 1 si la trame reçue est une notification du daemon local
- Pas d'usage pour le moment : 5^{th} et 4^{th} bits
- Chiffré: 3^{th} bit, égale 1 si la partie de données de la trame est chiffrée
- TCP utilisé: 2^{nd} bit, égale 1 si cette trame requit une garantie de livraison, à lire et assurer par la couche de transport
- Priorité : 1^{er} et 0^{th} bits, la priorité est parmi URGENT, HIGH, NORMAL, LOW, qui sera traitée par la couche transport

Une visualisation d'une trame SVC est donnée ci-dessous :

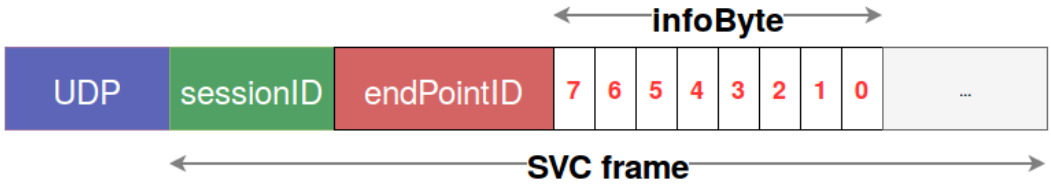

Figure 3.4 – La structure générale d'une trame SVC

Une trame de données du SVC contient, après l'entête, 4 octets de la longueur de la charge utile de données, suivie par la charge elle-même et le HMAC (l'algorithme et la longueur du HMAC dépendent de la version du SVC). Une trame de données doit être toujours chiffrée.

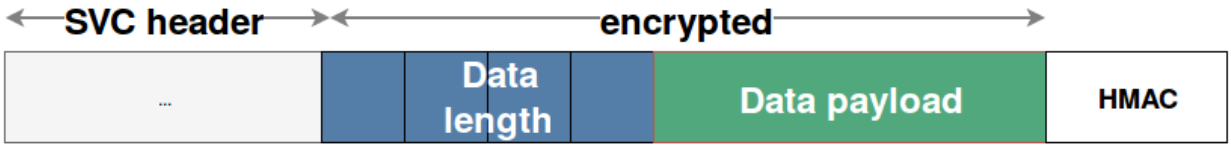

Figure 3.5 – Une trame de données du SVC

Une trame de commande du SVC commence par l'identité de commande (CID), suivie du nombre de paramètres (PRC). Le reste est la partie des paramètres, dans laquelle chaque paramètre est précédé par 2 octets de longueur. Sauf les commandes d'initialisation de connexion qui ne sont pas chiffrées, toutes les commandes restant sont chiffrées, et se terminent par un HMAC.

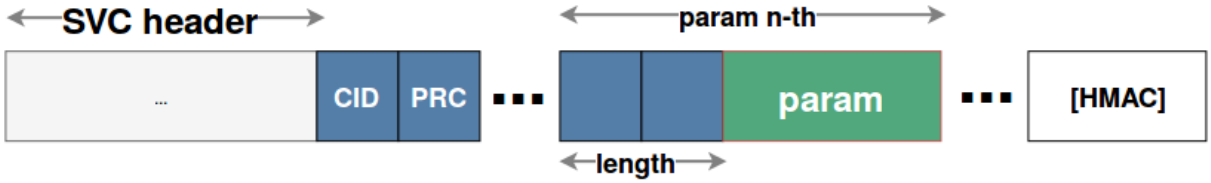

Figure 3.6 – Une trame de commande du SVC

4. Le détail d'un processus d'initiation de connexion

Une application crée une instance de SVC et demande la connexion. Dans la demande contient l'adresse de l'hôte à distance, qui sera utilisée par le « daemon » pour décider s'il doit créer un nouveau service (chaque service est en charge de connexion à un hôte unique).

À noter qu'il y a une seule instance du « daemon » qui s'exécute dans le background et qui est en charge de toutes instances du service. Pour pouvoir distinguer les demandes de connexion des applications différentes, le STEP_1 doit lui communique une identité de l'application (appID). Cette identité est liée à l'application et n'est pas au client sur lequel l'application s'exécute.

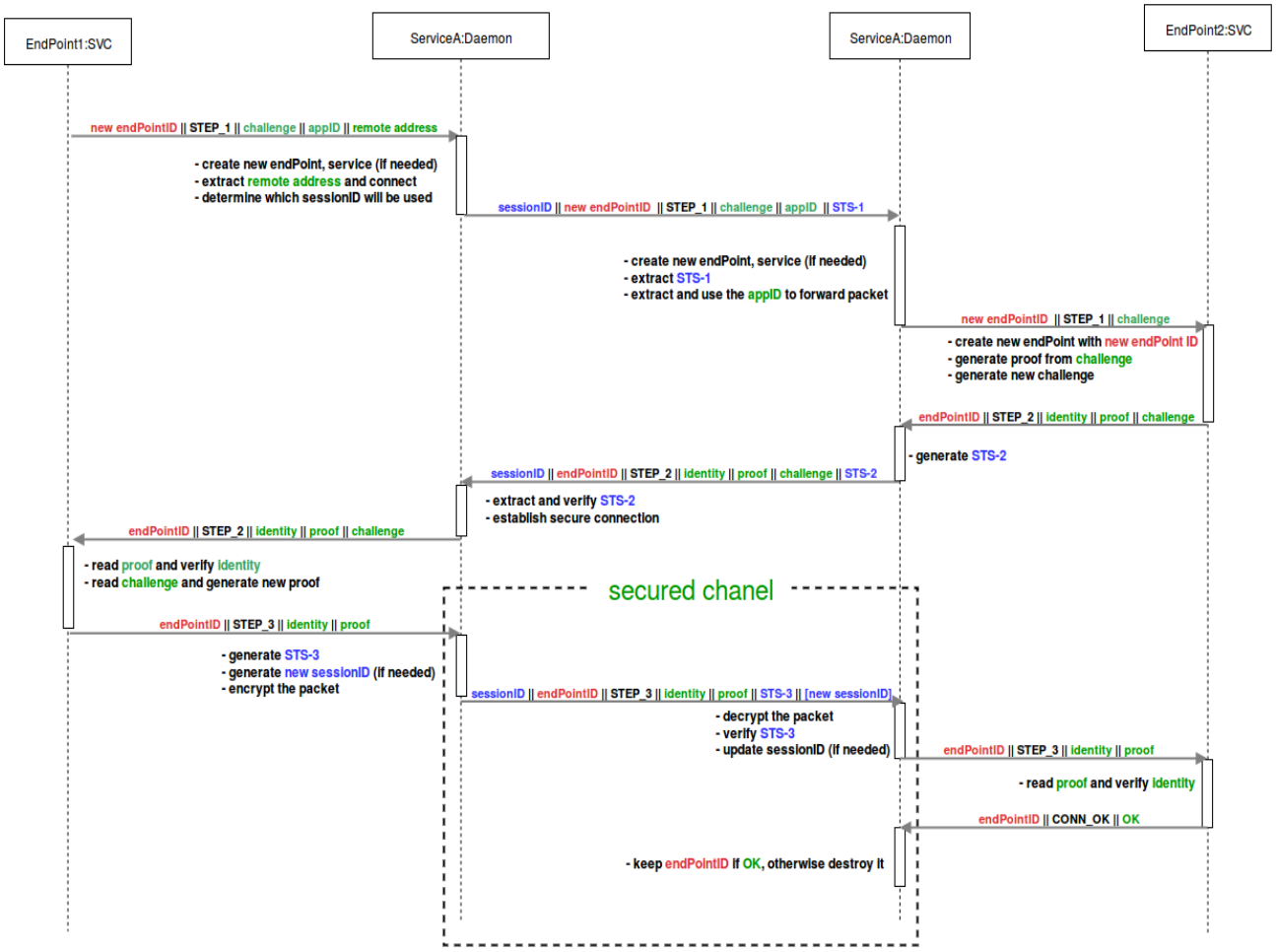

Figure 3.7 – Le processus d'initialisation de connexion du SVC

Une nouvelle identité de session est communiquée (si besoin) dans le STEP_3 pour identifier la session entre deux hôtes et qui servira à la reprise rapide de session.

Une demande de connexion et ses extrémités seront détruites si elle n'est pas réussie après un certain de temps.

IV. Conclusion

La phase de conception de cette solution est finie. Pendant la phase d'implémentation, des petites modifications peuvent être introduites, mais toujours en respectant les principes déterminés.

Une partie de l'implémentation est déjà en cours d'être réalisée. Dès qu'une version complète est disponible, on va tester et comparer sa performance avec les protocoles existants pour pouvoir l'améliorer. Une conception du HTP (Hybrid Transmission Protocol) et son implémentation sont aussi prévues dans nos prochains travaux suite à ce stage pour supporter le SVC.

Ce stage nous a permis, dans une approche « top-down », d'analyser les besoins de communications sécurisées dans une classe d'applications importante : celle des transports publiques. La collaboration du groupe ERISCS avec la RTM nous a permis une analyse précise des besoins dans un contexte opérationnel.

Nous avons pu proposer une architecture nouvelle optimisée du protocole SVC initialement proposé. Une implémentation expérimentale est faite pour pouvoir à la suite de ce stage évaluer les performances dans un contexte réel d'application par le partenaire industriel.

V. Références

[1] G. Risterucci, T. Muntean, L. Mugwaneza. “A new Secure Virtual Connector approach for communication within large distributed systems”, ERISCS Research Group.

[2] Intelligent garage and predictive maintenance, *http://www.ebsf2.eu/key-innovations/intelligent-garage-and-predictive-maintenance*

[3] Station-to-Station protocol, *https://en.wikipedia.org/wiki/Station-to-Station_protocol*

[4] Blake-Wilson, S.; Menezes, A. (1999), "Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol", *Public Key Cryptography*, Lecture Notes in Computer Science, **1560**, Springer, pp. 154–170