1 PROTECTION DES COMMUNICATIONS

Il y a plusieurs façons de définir une communication sécurisée. Ces définitions s'appuient sur un ensemble de propriétés de sécurité que l'on peut attendre d'une communication. Les principales propriétés de sécurité sont les suivantes : confidentialité, intégrité, authenticité, le déni plausible, la confidentialité persistante et la protection de l'identité lors des communications. La confidentialité permet d'éviter qu'un tiers non autorisé n'ait accès au contenu de la communication. L'intégrité assure que les messages échangés ne sont pas modifiés durant leur transfert. L'authenticité recouvre en partie l'intégrité et apporte en plus des garanties sur l'identité de l'émetteur d'un message. Le déni plausible donne essentiellement la possibilité à l'émetteur d'un message de nier en être l'auteur. La confidentialité persistante permet de garantir que même en cas de divulgation ultérieure des informations d'un utilisateur (clé cryptographique, certificats,...) il reste impossible de déchiffrer une communication passée. Il s'agit là d'une propriété de sécurité différente de la confidentialité, car elle concerne la protection a posteriori des messages échangées. La protection de l'identité est une forme d'anonymat. Elle peut recouvrir plusieurs notions. Nous parlons ici principalement de masquage d'identité, afin de ne pas divulguer d'informations sur l'origine de la communication sécurisée. Nous pouvons donc avoir plusieurs définitions de communication sécurisée, par exemple en considérant uniquement la confidentialité, ou en considérant à la fois confidentialité et authenticité.

Chacune des propriétés de sécurité induit un coût parfois important sur la mise en œuvre de communication sécurisée. En particulier, l'authenticité des communications nécessite un système de gestion de clés des utilisateurs. La gestion de clés peut être particulièrement complexe si l'on considère un grand nombre d'utilisateurs, ou un réseau de grande taille sur lequel il est difficile d'utiliser des canaux secondaires pour partager les clés. Ainsi, selon le type de réseau et d'utilisation, certaines propriétés de sécurité peuvent se voire exclues pour des raisons pratiques ou de performances. Nous considérons ici qu'une communication sécurisée nécessite l'application de toutes les propriétés de sécurité sans exception.

Il existe plusieurs façons de mettre en place un système utilisant des communi-

cations sécurisées. Les deux approches courantes sont l'utilisation d'un protocole de communication sécurisée et la mise en place d'un VPN [1]. L'utilisation directe d'un protocole de communication sécurisée permet d'établir, au niveau de l'application, une connexion sécurisée avec une autre entité (une autre instance de l'application, un serveur,...). Cette approche permet un contrôle fin au niveau de l'application du niveau de sécurité que l'on souhaite obtenir et facilite la maintenance des éléments de sécurité de l'application. En effet, en intégrant ces éléments de sécurité au niveau applicatif, il est possible de les maintenir à jour même lorsque les éléments extérieurs (bibliothèques de sécurité, fonctions du système d'exploitation,...) ne sont plus mis à jour.

La mise en place d'un VPN présente des caractéristiques différentes de l'utilisation directe d'un protocole de communication sécurisée mais fournit une sécurité similaire. En effet, un VPN peut se présenter de différentes façons, mais les communications effectives sont réalisées, dans le cas d'un VPN sécurisé, via un protocole de communication sécurisée "classique" par l'application VPN. La principale différence entre les deux approches est qu'un VPN peut permettre d'apporter des communications sécurisées à des applications n'ayant pas nativement cette possibilité. Comme un VPN est vu par les applications comme une interface réseau, n'importe quelle application communiquant sur un protocole supporté par la solution de VPN (typiquement du TCP [2] ou de l'UDP [3]) bénéficie de la sécurité fournie. Un point négatif de l'utilisation d'un VPN est qu'il n'est pas sous le contrôle des applications qui l'utilisent et peut donc présenter un niveau de sécurité inadéquat en fonction de certains usages. En particulier, une application n'a pas le moyen de savoir si un VPN est utilisé, car cet aspect est masqué par le système d'exploitation, ni le type de sécurité qu'il fournit le cas échéant ; certains VPN sont utilisés uniquement pour leur propriété de jonction de réseaux distants sans aucun élément de sécurité. Il est également plus complexe de mettre en œuvre une solution de type VPN, car elle nécessite l'installation et la configuration d'éléments tiers à une application. Cet aspect n'est pas en soit un problème de sécurité, mais apporte une contrainte de déploiement qui rend cette solution incompatible avec certaines classes d'applications. Un cas de situation incompatible est l'utilisation d'une application nomade, lancée sur des systèmes pour lesquelles l'utilisateur n'a pas accès à la configuration du système d'exploitation, rendant impossible la mise en place d'un VPN.

À partir de ces éléments, nous pouvons voir qu'il existe des contraintes à la

1. Virtual Private Network, réseau privé virtuel, réseau logique pouvant regrouper des machines situées sur différents réseaux physiques distants

2. Transmission Control Protocol, protocole de communications garantissant notamment l'ordre et l'unicité de la délivrance des messages

3. User Datagram Protocol, protocole de communications rapide n'ayant ni garantie de délivrance des messages, ni de garantie d'ordre d'arrivé

mise en œuvre de protocoles de communication sécurisée qui vont au-delà de l'application des propriétés de sécurités. Les contraintes d'exploitations doivent être prises en compte au plus tôt dans la conception d'un protocole afin que ce dernier soit adapté au besoin. Comme indiqué au paragraphe précédent, une contrainte basique est la simple possibilité de mise en place du protocole. Cette contrainte impose des restrictions sur les outils utilisables et leur déploiement. D'autres contraintes d'exploitations qui seront détaillées dans la suite du chapitre incluent les performances qui ne doivent pas nuire à l'expérience utilisateur, l'indépendance des outils utilisés vis-à-vis de composants non maîtrisés de l'application et la disponibilité sur différents systèmes notamment des systèmes mobiles.

Nous allons étudier dans ce chapitre des solutions permettant de mettre en place des systèmes de communication sécurisée répondant à la fois aux propriétés de sécurité proposées et aux contraintes d'exploitations. Ces dernières vont au-delà de la simple mise en place d'outils à chaque extrémité d'une communication. Cela implique que les outils de sécurité considérés ne peuvent pas dépendre d'éléments comme l'accès à la configuration avancée d'un système d'exploitation pour fonctionner. Pour cette raison, les solutions à base de VPN ne seront pas davantage détaillées.

Nous avons pour cela développé un protocole de communication sécurisée permettant d'établir une connexion sécurisée point à point entre deux systèmes. Les termes *client* et *serveur* sont utilisés ici pour désigner les deux extrémités d'une communication. Il ne s'agit pas de limiter les descriptions au modèle "un serveur, plusieurs clients". Ces termes sont utilisés ici pour différencier le système en attente de connexion (le *serveur*) du système initiant de façon active la connexion (le *client*). Ce chapitre couvre donc d'autres modèles, incluant les connexions point à point pour lesquels les rôles "serveur" et "client" restent présents.

L'annexe A présente un rappel rapide des outils cryptographiques auxquels il est fait référence tout au long du chapitre.

1.1 Protocoles de communications standards

Il existe actuellement plusieurs protocoles permettant de fournir certaines des propriétés de sécurité décrites dans ce chapitre. Nous nous intéressons ici aux protocoles permettant de sécuriser des communications réseau. Les trois outils les plus courants sont TLS [4], SSH [5] et IPSec [6]. Il existe d'autres protocoles spécialisés pour

4. Transport Layer Security, protocole de communication sécurisée remplaçant SSL

5. Secure Shell, protocole de communication sécurisée permettant notamment l'ouverture d'une invite de commande sur un système distant

6. Internet Protocol Security, protocole de chiffrement des échanges au niveau IP

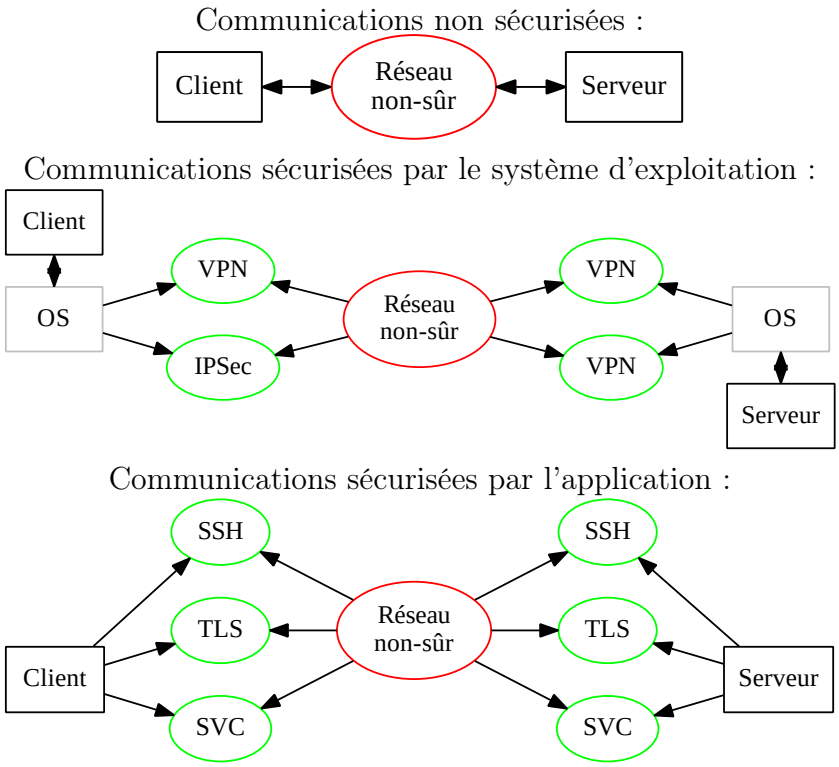

FIGURE 1.1 – Méthodes d'établissement de communication sécurisée

des usages spécifiques. Nous pouvons citer en exemple le protocole ZRTP [7] dédié au chiffrement des communications VoIP [8], ainsi qu'OTR [9] dédié aux échanges de messages instantanés. Ces deux solutions sont très spécifiques à des problèmes particuliers et ne seront pas étudiées plus en détail dans ce document.

Du point de vue de l'utilisation, les protocoles et outils que sont TLS, SSH et IPSec ont tous un fonctionnement similaire : ils permettent l'établissement d'une connexion sécurisée, utilisée par la suite pour l'échange de nombreux messages.

Les solutions construites sur IPSec sont, comme les VPN, incompatibles avec les contraintes d'exploitations, notamment la possibilité de mise en œuvre en mobilité. La figure 1.1 présente deux façons générales d'établir des communications sécurisées. Seule des approches ne nécessitant pas la participation active du système d'exploitation sont considérés, afin de rester compatible avec les contraintes d'exploitations. C'est pour cela qu'IPSec comme les VPN ne seront pas davantage détaillés ici.

7. Z-Real-time Transport Protocol, protocole de protections des communications VoIP
8. Voice over IP, voix sur IP, communications audio circulant sur des réseaux informatiques basés sur le protocole IP
9. Off-the-record, confidentiel

Nous ne considérons que les protocoles TLS et SSH qui sont les protocoles les plus utilisés pour sécuriser les communications au niveau applicatif. Dans leur fonctionnement minimum, TLS et SSH permettent d'assurer la confidentialité des échanges et l'authenticité du serveur. Au-delà de ces fonctionnalités minimales, ils permettent sous certaines conditions d'assurer des propriétés d'authenticité du client. Dans la suite du chapitre, nous présentons en détail ces deux protocoles. En particulier sont décrits les éléments qui réduisent leur efficacité pour certaines classes d'applications (communications internes à une application, voir section 1.2.3), qu'il s'agisse d'affaiblir la sécurité ou d'impacter négativement les performances du système.

1.1.1 TLS

TLS est actuellement le protocole le plus utilisé pour sécuriser les communications au niveau applicatif sur réseau TCP/IP. Il fait suite au protocole SSL [10], publié pour la première fois par Netscape[24]. Deux versions de SSL ont été exploitées : SSL2.0 et SSL3.0. Bien qu'initialement basé sur SSL, TLS n'est pas strictement compatible avec ces dernières. Les deux objectifs principaux de TLS sont la sécurité des communications et l'interopérabilité entre implémentations, comme cela est décrit dans la RFC 5246[14]. La sécurité des communications fournie par TLS est basée sur des opérations cryptographiques incluant chiffrement, échange de clés et signatures numériques pour l'authentification. L'interopérabilité est fournie par une série d'étapes de négociation permettant d'une part de déterminer les capacités cryptographiques des deux systèmes cherchant à établir une connexion sécurisée et d'autre part à échanger dans un format standardisé les informations d'identification.

Pour fonctionner lors d'interaction avec des tiers inconnus, l'interopérabilité nécessite le support d'anciens protocoles (notamment SSL et les premières versions de TLS). En effet, l'utilisation de ces différents protocoles que ce soit du côté des applications (clients comme serveur) est toujours très répandue, comme indiqué dans la table 1.1. Les premiers chiffres présentés sont issues de l'observation d'un million de sites "populaires" début 2014[67]. Sur ce million, 45,1% disposaient d'un accès TLS (soit 451 470 serveurs). Les valeurs plus récentes proviennent du site SSLLabs[46] et sont produites à partir des résultats de tests de près de 200 000 serveurs visant à évaluer leur niveau de support du protocole TLS. Si un net recul des anciennes versions du protocole (versions non sûres) est visible, elles restent toutefois très présentes et restent un vecteur d'attaque important. De même, les versions plus récentes restent sous-représentées, entre autre à cause de l'inertie liée au standard.

10. Secure Socket Layer, protocole de communication sécurisée remplacé par TLS

Version	Date de création	Disponibilité début 2014	Disponibilité mi-2015	
SSL2.0	1995	18,9%	10,8%	(-8,1)
SSL3.0	1996	99,6%	33,8%	(-65,8)
TLS1.0	1999	98,9%	99,2%	(+0,3)
TLS1.1	2006	32,2%	64,3%	(+32,1)
TLS1.2	2008	33,2%	66,5%	(+33,3)

TABLE 1.1 – Disponibilité des différentes versions de SSL/TLS

Disponibilité de chaque versions pour 451 470 serveurs permettant l'utilisation des protocoles SSL et TLS pour les chiffres de 2014, et sur près de 200 000 serveurs pour les chiffres de 2015.

1.1.1.1 Sécurité

TLS fournit la confidentialité des échanges entre des clients et un serveur, ainsi que l'authentification de ce dernier. Lorsque la négociation TLS aboutit le client a la garantie qu'il est bien connecté à un serveur légitime et que leur échange est confidentiel. Nous pouvons alors parler de connexion sécurisée. Il est également possible d'utiliser le protocole TLS pour authentifier le client. Dans les deux cas, l'authentification est basée sur un système de clé publique utilisant des certificats X.509[9]. L'authentification du serveur peut être réalisée par le client sans échange préalable, en s'appuyant sur un système de tiers de confiance connus du poste client. Il faut noter que dans tous les cas, les informations d'identité, c'est-à-dire les certificats, sont échangée sans chiffrement. Cela inclut le certificat du client si l'authentification via TLS est utilisée. TLS ne fournit donc pas de confidentialité sur l'identité des utilisateurs, mais uniquement sur le contenu des échanges.

Les protocoles SSL et TLS fonctionnent sur un schéma similaire : une phase de négociation permet d'établir les paramètres de sécurité de l'échange et une phase d'exécution met en œuvre ces paramètres.

Phase de négociation : La phase de négociation implique deux éléments : le choix des algorithmes à utiliser et l'exécution de l'échange de clés. Le choix des algorithmes se fait sur la base des informations échangées entre le client et le serveur : le client annonce la liste d'algorithmes qu'il supporte et le serveur choisit dans cette liste ceux qu'il supporte et qui présentent le meilleur niveau de sécurité. En théorie, cela permet de garantir un niveau de sécurité minimale depuis les deux extrémités de la connexion. En pratique, il existe encore de nombreux clients et serveurs dépendant d'algorithmes vulnérables, qui imposent par inertie de conserver leur support. La table 1.2 présente les principaux algorithmes supportés par les différentes versions de SSL et TLS. Nous constatons que les plus anciennes versions ne supportent que peu (ou pas) d'algorithmes permettant d'apporter la

Version du protocole	Échange de clés (sans conf. persist.)	Échange de clés (avec conf. persist.)
SSL2.0	RSA	-
SSL3.0	DH-(RSA/DSS)	DHE(RSA/DSS)
TLS(v1, 1.1, 1.2)	ECDH-(RSA/DSS),	ECDHE(RSA/DSS)
	PSK, PSK-RSA, SRP-(RSA/DSS)	(DHE/ECDHE)-PSK

TABLE 1.2 – Algorithmes d'échange de clés de TLS

DH :Diffie-Hellman, DHE :Ephemeral Diffie-Hellman, ECDH :Elliptic-Curve DH, PSK :Pre-Shared Key, SRP :Shared Password, conf. persis. : Confidentialité persistante

confidentialité persistante. De manière générale il est déconseillé d'utiliser SSL. Pour assurer un niveau de sécurité acceptable il faut déployer au minimum des solutions basées sur TLS version 1.1. Finalement, bien que TLS permette une négociation sûre basée sur un échange de clés de type ECDH [11] fournissant à la fois un haut niveau de sécurité et une confidentialité persistante, leur usage reste optionnel et il est toujours possible d'utiliser des échanges de clé plus faibles en termes de propriétés de sécurité.

Chiffrement : Une fois la phase de négociation complétée, le protocole TLS est utilisé pour chiffrer les échanges afin d'apporter la propriété de confidentialité. Le chiffrement est réalisé à partir des paramètres négociés, dont le choix de l'algorithme à utiliser. TLS supporte différent algorithmes de chiffrement, qui peuvent être utilisés de différentes façons, selon les versions du protocole. Certains algorithmes sont maintenant considérés comme vulnérables. En particulier l'utilisation du mode CBC [12] par SSL et les premières versions de TLS n'étaient pas faites de façon sécurisée. Les algorithmes de chiffrement supportés par SSL et les premières versions de TLS incluent des algorithmes considérés comme non sûres comme DES [13] ou RC2 [14]. Comme montré dans la table 1.3, seule la version 1.2 de TLS permet d'utiliser autre chose que le mode CBC pour les opérations de chiffrement. L'utilisation en place de ce mode de chiffrement est à la base de plusieurs attaques sur différentes versions de TLS.

11. Elliptic-Curve Diffie-Hellman, Diffie-Hellman basé sur courbes elliptiques
12. Cipher Block Chaining, enchaînement de blocs chiffrés : mode d'utilisation consistant à faire intervenir pour chaque bloc chiffré l'état du bloc précédent, afin de construire une chaîne
13. Digital Encryption standard, ancien algorithme de chiffrement par blocs
14. Rivest Cipher 2, algorithme de chiffrement par bloc

Version du protocole	Algorithmes supportés	Mode de chiffrement
SSL(2.0,3.0)	3DES,IDEA,DES,RC2,RC4	CBC
TLS1.0	+(AES,Camellia,ARIA)	identique
TLS1.1	-RC2	identique
TLS1.2	-(IDEA,DES)	+(GCM,CCM)

TABLE 1.3 – Algorithmes de chiffrement de TLS

Le mode de chiffrement s'applique uniquement aux chiffrements par blocs. Les lignes concernant TLS montrent les ajouts/suppressions par rapport à la ligne précédente.

1.1.1.2 Fonctionnement

TLS agit comme une couche intermédiaire entre l'application et la connexion TCP/IP utilisée pour la communication. L'exemple de fonctionnement le plus connu de TLS est le protocole HTTPS [15][56] utilisé pour transporter des requêtes HTTP [16] de façon sécurisée. Dans cet exemple, des requêtes HTTP non sécurisées sont encapsulées par le protocole TLS afin de sécuriser les échanges. Le traitement au niveau applicatif, que ce soit côté client ou côté serveur, a lieu indépendamment du protocole TLS sur les requêtes après leur déchiffrement. Le protocole TLS dégage l'application de tous les aspects de sécurité de la communication en assurant de façon quasi transparente l'authentification et le chiffrement. La seule contrainte sur l'utilisation du protocole TLS est la disponibilité de composants logiciels permettant de l'implémenter, qu'il s'agisse du protocole lui-même ou des algorithmes cryptographiques utilisés.

TLS est également utilisé comme élément de sécurité pour les VPN. La fonction de base d'un VPN est de créer un réseau logique virtuel pouvant tourner sur plusieurs réseaux différents, afin d'offrir une continuité réseau à des machines distantes. Comme pour HTTPS, les communications de base VPN ne sont pas sécurisées. Ces communications peuvent en revanche être encapsulées dans différents protocoles de sécurité, dont TLS, afin de fournir une connexion sécurisée.

1.1.1.3 Faiblesses

Bien que la dernière version du protocole TLS, la version 1.2, apporte des solutions à la plupart des problèmes des versions précédentes, il existe des faiblesses liées à la construction du protocole lui-même qui ne peuvent pas être corrigés sans en modifier le fonctionnement. D'autres vulnérabilités sont liées à un manque de contrôle de l'environnement d'utilisation du protocole TLS qui dépend de nom-

15. Hypertext Transfer Protocol Secure, version de HTTP sécurisée par TLS
16. Hypertext Transfer Protocol, protocole utilisé pour le transfert de pages Web

breux éléments extérieures comme la configuration du système d'exploitation ou l'utilisation d'une bibliothèque extérieure pas nécessairement maintenue à jour.

Algorithmes vulnérables : Le point faible le plus exploitable des protocoles SSL et TLS est l'utilisation d'algorithmes cryptographiques vulnérables, ou de combinaisons d'algorithmes introduisant de nouvelles vulnérabilités. Comme indiqué précédemment, ces algorithmes sont toujours largement utilisés. Certains algorithmes en particulier ont posé ou posent d'importants problèmes. Le premier, RC4 [17], est considéré comme vulnérable. Son utilisation est interdite dans toutes les versions de TLS[52]. Malgré cela, il y a toujours des clients et des serveurs en exploitation qui utilisent cet algorithme : 56,1% de serveurs supportent RC4 d'après les résultats de 2014 d'une enquête permanente[46]. Le deuxième problème majeur dans les algorithmes mis en œuvre par TLS est l'utilisation incorrecte du mode CBC. Ce mode d'utilisation, comme beaucoup, nécessite un vecteur d'initialisation pour fonctionner. Bien que non secret, ce vecteur d'initialisation doit être unique et imprévisible pour chaque utilisation, sans quoi de nombreuses attaques peuvent être appliquées. Dans le cas des versions vulnérables (SSL2.0, SSL3.0, TLS1.0), le vecteur d'initialisation est prévisible et peut permettre de récupérer le contenu confidentiel de la communication. Pour finir sur ce point, les algorithmes utilisés pour assurer l'intégrité des données, de type HMAC [18], sont basés sur des fonctions de hachage faibles : soit MD5 [19], soit SHA1 [20] pour toutes les versions de TLS avant la version 1.2. Seule cette dernière inclut de nouveaux algorithmes comme SHA2 [21]. La faiblesse du contrôle d'intégrité peut permettre l'altération des données sans que cela ne soit détecté. Combiné à d'autres faiblesses, il devient alors possible de falsifier des messages apparemment légitimes. L'utilisation de ces algorithmes peut à elle seule rendre inutile la protection fournie par TLS en rendant le chiffrement inefficace ou en permettant de falsifier des messages.

Rétro-compatibilité : Le protocole TLS permet de revenir à d'anciennes versions, y compris à des versions de SSL. Cette rétro-compatibilité permet de conserver l'interopérabilité dans le cas où un client se connecterait à un serveur ne disposant pas des dernières versions du protocole ou vice versa. Cependant, cette interopérabilité permet des attaques sur TLS qui forcent la négociation à utiliser une version vulnérable du protocole afin d'exploiter des failles connues et corrigées[49]. Nous pouvons citer la faille POODLE, qui exploite une faiblesse de

17. Rivest Cipher 4, algorithme de chiffrement à flot
18. Hash-based Message Authentication Code, code d'authentification de message basé sur des fonction de hachage
19. Message Digest 5, fonction de hachage simple
20. Secure Hash Algorithm 1, fonction de hachage
21. Secure Hash Algorithm 2, fonction de hachage succédant à SHA1, plus sûr que ce dernier

SSL version 3, et qui pouvait impacter des systèmes disposant de versions plus récentes, grâce à la rétro-compatibilité. De manière générale, les recommandations sur l'interdiction d'anciens algorithmes et les changements dans leurs modes d'utilisation, ne s'appliquent pas aux anciennes versions du protocole. Pour réellement renforcer la sécurité des communications basées sur TLS, il a fallu limiter cette rétro-compatibilité en interdisant de descendre à un niveau inférieur à TLS1.0[6].

Environnement d'utilisation : Les problèmes liés à l'environnement dans lequel est utilisé le protocole TLS peuvent compromettre la sécurité à plusieurs niveaux. L'utilisation d'une PKI [22] gérée par un tiers (en général le système d'exploitation) peut faciliter des attaques de type MitM [23] ou d'usurpation d'identité[43]. En effet, l'authentification du serveur est basée sur la validation de son certificat. Cette validation dépend d'un magasin de certificats dit "de confiance" stocké sur la machine cliente et géré le plus souvent par le système d'exploitation. L'injection d'un certificat par un attaquant dans ce magasin compromet l'élément de confiance qu'il fournit, rendant invalide son utilisation pour l'authentification du serveur et donc, cassant la sécurité fournie par TLS.

Un autre problème lié à l'environnement d'utilisation de TLS est en rapport avec l'origine de l'implémentation utilisée. Certains systèmes d'exploitation fournissent directement leur implémentation de TLS. Cela permet un déploiement plus simple, mais associe la sécurité de l'application à la sécurité fournie par le système d'exploitation. Au-delà du risque posé par l'utilisation d'un élément critique provenant d'une source non contrôlée, il existe un problème bien plus important : celui des mises à jour du système d'exploitation. En effet, obtenir des mises à jour de sécurité pour l'ensemble du système devient obligatoire pour contrer de nouvelles vulnérabilités. Cependant, certains systèmes d'exploitation ont une durée de maintenance très courte et ne peuvent rapidement plus être mis à jour. Cela n'empêche pas leur utilisation pendant cette période de "fin de vie". Le déploiement d'applications sur des environnements mobiles tels que les smartphones est particulièrement concerné par cette restriction, car le plus souvent il est très difficile (parfois même impossible) d'obtenir des mises à jour pour des appareils encore couramment utilisés. Dans ces conditions, les applications déployées pour de telles systèmes et dépendantes de composants logiciels obsolètes peuvent devenir vulnérable au fil de la découverte de nouvelles vulnérabilités, sans possibilité de corrections.

22. Public Key Infrastructure, infrastructure à clé publique : ensemble de systèmes permettant la gestion de certificats numériques de clé publiques

23. Man in the Middle, l'homme au milieu : intercepter et/ou modifier les communications entre deux systèmes

Fonctionnalités superflues : Certaines fonctionnalités de TLS sont parfois superflues et ont pu servir de vecteur d'attaque pour affaiblir la sécurité des communications. Parmi ces fonctionnalités nous comptons des optimisations de renégociation et une fonctionnalité de "battement de cœur" (ping). La renégociation devait permettre d'économiser des ressources dans le cas ou un même client devait établir plusieurs connexions successives avec un même serveur. Le fonctionnement est le suivant : les deux extrémités de la communication retiennent les paramètres négociés, associé à un numéro de session. Lors d'une nouvelle connexion, un client peut envoyer son numéro de session. Si le serveur retrouve cet identifiant, la session précédente est réutilisée. Cette fonctionnalité permet dans certaines conditions à un attaquant d'injecter un message dans l'échange en se faisant passer pour le client initial ou de complètement intercepter une communication sécurisée. L'autre fonctionnalité ayant servi à attaquer le protocole est le "heartbeat", ou "battement de cœur". Cette fonction, similaire à un "ping", sert essentiellement à s'assurer que le serveur est toujours actif. En soi, elle ne présente pas de problème particulier, mais une vulnérabilité logicielle a été trouvée dans la bibliothèque OpenSSL, l'une des implémentations les plus répandues. Cette vulnérabilité permet à un attaquant de récupérer des blocs arbitraires de la mémoire du serveur et ce sans laisser de traces. Parmi les données ainsi exposées il est possible de trouver des informations sur les utilisateurs, mais surtout la clé privée utilisée par le serveur. La compromission de cette dernière rend possible la mise en place par un attaquant d'un serveur usurpant l'identité du serveur initial. Puisque la fonctionnalité de base "heartbeat" est peu utilisée en pratique, cette vulnérabilité se trouve être une illustration intéressante du fait que plus un protocole est "large" en application, plus sa surface d'attaque est importante. Cette faille est connue sous le nom de "heartbleed"[15, 27]. Dans les deux cas, la solution à court terme a été de désactiver la fonctionnalité mise en cause, ce sans perturbation majeure à l'utilisation. En effet, la renégociation de TLS, bien que coûteuse, n'a pas un impact très important sur des systèmes correctement dimensionnés et la faille "heartbleed" exploitant une fonction peu utilisée, sa désactivation n'a pas posé de problèmes. Ces deux exemples ont fait l'objet de corrections par la suite, sous la forme d'une renégociation plus sûre pour TLS[57] et sous la forme d'un correctif logiciel pour OpenSSL concernant "heartbleed".

En plus des problèmes évoqués, il existe aussi des problèmes liés à la façon d'utiliser certains algorithmes. Des attaques existent, dont certaines sont encore applicables à TLS version 1.2[62, 60].

En dehors des problèmes de vulnérabilités et d'attaques sur l'utilisation du protocole, il y a un élément du protocole qui s'avère être un point faible dans le contexte des communications sécurisées envisagé dans ce chapitre. Les informations d'identité, qu'il s'agisse du serveur ou du client, sont échangées sans aucune protection. Il n'y a lors de cette étape aucun chiffrement mis en œuvre. De manière

générale, si une connexion avec un serveur “fixe” est établie, dévoiler l'identité de ce système n'est pas un problème majeur.

En revanche, si l'authentification du client via TLS est utilisée, l'information d'identité du client est transmise et peut permettre d'identifier les utilisateurs d'un service de façon fiable. Cela pose un problème pour la protection de la vie privée et ne peut pas être corrigé dans la version actuelle de TLS. Afin d'éviter ce problème, toute application utilisant TLS et souhaitant authentifier ses clients doit mettre en place un mécanisme secondaire d'authentification opérant via la connexion sécurisée par TLS. Cette approche est contraignante ; en effet si l'authentification de l'utilisateur doit se faire sur une connexion sécurisée, cela signifie terminer l'ensemble de la négociation dans un premier temps et authentifier le client ensuite. Dans le cas ou l'authentification échouerait à ce niveau, les ressources utilisées pour l'établissement de la connexion sécurisée ont déjà été consommées. Cette utilisation superflue de ressources tant sur le serveur que sur le client peut mener à une autre forme d'attaque : le DoS [24].

1.1.2 SSH

Le protocole SSH est largement et principalement utilisé pour interagir avec des systèmes distants via l'ouverture d'une invite de commande au travers d'une connexion réseau, mais ce n'est pas là son seul usage. En fait, SSH est la combinaison de trois éléments qui lui permettent d'agir comme une couche de transport sécurisé pour différents types d'applications[72].

Le premier élément est appelé la couche transport[74]. Cet élément fournit des communications bidirectionnelles sécurisées entre le client et le serveur, ainsi que l'authentification du serveur. Cette authentification est basée sur la machine hôte et identifie donc le système plutôt qu'un utilisateur ou un service. Elle peut se faire de plusieurs façons, mais le plus souvent une procédure d'authentification à clé publique est utilisée. Lors de la première connexion, cette clé est récupérée par le client puis validée et retenu pour les connections ultérieures afin de ne pas avoir à répéter le processus de validation. La couche transport de SSH permet aux autres composants de bénéficier d'une connexion sécurisée, car elle fournit le chiffrement pour la confidentialité ainsi que l'intégrité des échanges.

Le deuxième élément du protocole est le protocole d'authentification[71]. Il s'exécute avec le support de la couche transport sécurisé, ce qui permet de garder confidentielles les informations d'identification de l'utilisateur. Le protocole d'authentification de SSH fournit plusieurs méthodes d'authentification, dont des mécanismes basé sur un système challenge/réponse et des signatures à clé publique.

24. Denial of Service, déni de service : empêcher le fonctionnement normal d'un service en en bloquant l'accès ou les processus

Il inclut également des méthodes basiques comme la transmission des identifiants en texte clair, mais ces dernières sont plus rarement utilisées.

Le troisième et dernier élément du protocole SSH est le protocole de connexion[73]. Il permet de réaliser des communications au niveau applicatif grâce à la gestion de différents canaux pouvant être utilisés pour différents types d'informations. Ces canaux sont ensuite multiplexés de façon transparente sur la couche transport de SSH. Il peut y avoir par exemple un canal pour l'utilisation d'une invite de commande et un second canal pour réaliser du routage IP de façon sécurisée entre le client et le serveur. Comme ces différents canaux utilisent la couche transport pour leur communication effective, ils sont automatiquement rendus confidentiels et leur intégrité est vérifiée.

Il existe aujourd'hui peu de faiblesses connues sur le protocole SSH. La plupart des vulnérabilités datent de la première version du protocole[11] et ont été rendues obsolète par la seconde version (SSH-2). L'adoption quasi universelle de la deuxième version du protocole fait qu'il est acceptable et recommandé de complètement désactiver le support pour la version initiale, ce qui évite les problèmes liés à la rétrocompatibilité. Comme pour TLS, il peut persister des faiblesses liées à l'utilisation d'algorithmes de chiffrement faibles[3]. De la même manière, la technique permettant de réduire l'impact de ces faiblesses consiste à retirer les algorithmes faibles de l'étape de négociation. Un autre problème du protocole SSH est la difficulté à l'intégrer dans certaines applications. L'implémentation la plus populaire (OpenSSH) est conçue pour fonctionner en tant que processus indépendant, ce qui n'est pas toujours possible. D'autres implémentations existent, mais elles posent pour la plupart des problèmes de portabilité ou de licences. Il s'agit principalement de produits commerciaux dont la disponibilité est limitée à certaines architectures.

1.1.3 Autres approches

Comme indiqué précédemment, il existe d'autres types de solutions pour établir des communications sécurisées que l'utilisation d'un protocole de communication sécurisée. Parmi les exemples donnés, nous considérons l'utilisation de VPN et IPSec, ainsi que les protocoles comme OTR. Il existe également des approches moins standardisées, mais s'approchant toujours d'une solution existante. Certains systèmes permettent, de façon similaire au protocole OTR, d'envoyer des messages hors-ligne à un destinataire connu à l'avance, via un système à clé publique. Ces approches ne sont pas nécessairement moins sécurisées et certaines peuvent fournir l'ensemble des propriétés de sécurité décrites ci-dessus. Cependant, elles présentent des contraintes non pas de sécurité, mais d'utilisation, qui les rendent incompatibles avec le modèle considéré. C'est pour cela que cet état de l'art s'est concentré sur les protocoles répondant aux problématiques de connexion sécurisée dans un contexte compatible avec le déploiement d'applications sur systèmes variés.

1.2 Types de communications

Des clients lourds aux applications basées sur le cloud computing, les logiciels modernes dépendent des communications réseaux pour fonctionner. Ces communications peuvent contenir des informations confidentielles ou sensibles et nécessitent donc une sécurité adaptée.

Il y a plusieurs approches pour catégoriser les communications ; par exemple basé sur le contenu (privée/publique), le volume de données ou la latence. Ici nous considérons une contrainte de nature différente. Nous séparons les communications en deux types distincts : les communications dites "externes" ou interopérable et les communications "internes". Les communications externes regroupent les communications établies entre applications différentes. Un exemple pour ce type de communication est l'ouverture d'une page web par un navigateur communiquant avec un serveur HTTP. Les deux logiciels concernés ici, le client et le serveur, peuvent être totalement indépendant l'un de l'autre. Pour que ce type d'échanges fonctionne, il est nécessaire d'avoir un protocole de communication permettant l'interopérabilité, en l'occurrence le protocole HTTP. Les communications internes regroupent les échanges réalisés entre les instances d'une même application. Cela recouvre entre autre le schéma classique d'une application client/serveur, pour laquelle le logiciel client et le logiciel serveur font partie d'une unique application. Dans ce type de communications, l'interopérabilité n'est pas requise, car la même entité a le contrôle des deux extrémités lors des échanges.

Les protocoles de communication sécurisée TLS et SSH correspondent au modèle externe. Même s'il est possible de les utiliser pour les communications internes ils disposent de nombreux éléments qui deviennent alors redondants. Ces éléments, comme une phase de négociation complexe, sont des vecteurs d'attaques potentiels. Dans ce chapitre, nous nous intéressons au cas des communications internes et ce pour deux raisons. Tout d'abord, nous avons pu voir qu'il existait de nombreux protocoles existants pour le cas des communications externes, alors qu'il n'en existe pas pour le cas des communications internes. Cela laisse un vide qu'il est utile de combler. Ensuite, le modèle des communications internes permet d'avoir plus de latitude vis-à-vis des contraintes qui peuvent être imposées à l'utilisation. En particulier, nous pouvons dans ce cas présumer du support de certains algorithmes ou de l'existence d'une solution de mise à jour unifiée applicable aux différents composants intervenant dans un échange. Ces contraintes supplémentaires permettent de réduire considérablement le nombre de situations attendues lors de la négociation d'une communication sécurisée. Une conséquence de cela est qu'il devient possible de développer un protocole de communication sécurisée pour lequel l'étape de négociation est simplifiée sans faire de compromis sur les propriétés de sécurité fournies.

1.2.1 Contraintes communes

Qu'il s'agisse du cas "externe" ou "interne" l'ensemble des applications communicantes partagent des contraintes communes : performances, fiabilité du support de communication et sécurité du système d'exploitation.

Performances : Il y a plusieurs types de métriques lorsque l'on parle de performances des communications comme la bande passante, la latence, la gigue, le nombre d'échanges, le nombre de sauts, etc. Dans le contexte des protocoles de communications au niveau applicatif, nous nous intéressons aux métriques sur lesquelles l'application peut avoir une influence : la bande passante, la latence et le nombre d'échanges. Nous pouvons aussi ajouter aux métriques de performances réseau le coût en ressources processeurs et mémoire imposé sur les systèmes communicants par le protocole de communication. Un protocole de communication sécurisée impacte négativement toutes ces métriques. Afin d'établir une connexion sécurisée, un protocole de communication sécurisée doit ajouter des données permettant de mettre en œuvre au moins le chiffrement. Ces données additionnelles peuvent faire l'objet d'une négociation initiale, ou de données ajoutées à chaque message. L'ajout d'autres propriétés de sécurité (comme l'intégrité) impose aussi d'ajouter davantage de données à un échange. Ces éléments additionnels augmentent le besoin en bande passante de l'application et dans le cas d'une négociation initiale, augmentent le nombre d'échanges nécessaires sur le réseau. L'utilisation d'algorithmes complexes, notamment d'algorithmes cryptographiques, impose un coût en temps de calcul qui s'ajoute au temps de traitement de l'application. Ce temps de calcul impacte lui aussi négativement les performances réseau, car il peut augmenter la latence de l'application.

Fiabilité du support de communication : La sécurité d'une communication ne se limite pas à la confidentialité. Parmi les propriétés de sécurité décrites au début de chapitre, l'intégrité et l'authenticité des communications sont essentielles. Nous considérons deux causes possibles de falsification de message lors d'un échange sécurisé : cause accidentelles et malveillance. Le cas de la malveillance est clair : elle est du fait d'un attaquant cherchant à modifier le contenu d'une communication. Les causes accidentelles peuvent être plus nombreuses. Les différents supports utilisés pour les communications numériques pouvant introduire des erreurs accidentelles, il est essentiel de pouvoir détecter ces cas, même si certains types de support (par exemple les communications TCP/IP) ont une garantie d'intégrité des messages. Notons que cette garantie est là pour détecter les altérations accidentelles et ne protège pas contre la malveillance. Dans les deux cas, du point de vue d'un protocole de communication sécurisée, il est primordial de pouvoir garantir de façon fiable et prouvable l'intégrité d'un message. Cela prémunit

à la fois contre la malveillance et les accidents de communication.

Sécurité du système d'exploitation : La nature modulaire des systèmes modernes a conduit à l'intégration de nombreuses fonctionnalités au niveau du système d'exploitation, que ce soit sous la forme d'une API[25] spécifique ou sous la forme de bibliothèques logicielles embarquées. Comme exemple, nous pouvons citer le déploiement de bibliothèques logicielles gérées par différentes distributions Linux, l'utilisation de fonctions cryptographiques de systèmes d'exploitation tels que Windows ou MacOS ou encore l'intégration de bibliothèques dans le système d'exploitation Android. La fourniture par le système d'exploitation de composants de sécurité introduit une dépendance entre la mise à jour générale du système d'exploitation et la mise à jour de ces bibliothèques. Cette dépendance devient un problème lorsque l'on se place dans un contexte de sécurité, en particulier pour les bibliothèques de cryptographie. En effet, dans le monde de la sécurité, les mises à jour logicielles sont parfois nombreuses et doivent être rapidement déployées pour contrer des attaques dites "zero day". Ces dernières mettant en œuvre des vulnérabilités dès leur découverte (parfois même avant leur publication), la réactivité dans le déploiement de correctifs est essentielle. C'est dans ces conditions que la dépendance au système d'exploitation pour des services de sécurité est un problème qu'il s'agisse de délai avant le déploiement de correctifs, ou plus simplement de l'absence de mises à jour, car le cycle de vie du système d'exploitation est considéré comme terminé par son fournisseur. Cette dépendance peut rendre vulnérable une application. Ce problème n'est pas directement dû à la construction d'un protocole de communication sécurisée, mais il peut être limité dans sa spécification. Des recommandations sur la façon d'intégrer et d'utiliser un protocole sont essentielles pour éviter cette situation.

1.2.2 Communications interopérables

Le cas des communications externes, ou "interopérable", nécessite que des logiciels provenant de différentes sources puissent communiquer. Ces logiciels peuvent utiliser des versions différentes d'un même protocole, ou des versions identiques, mais supportant des ensembles différents d'algorithmes ou d'options. Pour permettre cela, il est nécessaire de supporter au niveau du protocole des options de rétro-compatibilité et de négociations d'options. Dans ces conditions, le développeur d'une application ne peut faire que des suppositions faibles sur les algorithmes qui seront supportés par l'autre extrémité d'une communication. Ces éléments de rétro-compatibilité et de sélection d'options élargissent la surface d'attaque pos-

25. Application Programming Interface, Interface de programmation exposant les fonctions d'un système ou d'une bibliothèque logicielle

sible. Les protocoles qui répondent à ce modèle font un compromis entre la disponibilité et la sécurité, au détriment de cette dernière.

La plupart des protocoles de communication sécurisée, en particulier TLS et SSH se placent dans ce cadre. Comme les communications inter applications peuvent être une nécessité, ces protocoles restent une réponse acceptable dans cette situation. Cependant, ils nécessitent une attention particulière à l'utilisation. En particulier, TLS est la réponse "classique" mais son utilisation doit prendre en compte tous les points faibles évoqués plus haut.

Il est possible de mettre en place un ensemble de recommandations qui permettent d'utiliser TLS avec un bon niveau de confiance. Le protocole SSH ne présente pas les mêmes risques dans une utilisation générique et ne nécessite donc pas de précautions particulières, au-delà des considérations usuelles lors de la mise en place de connexion sécurisées.

Utilisation sûre de TLS : Dans le cas où la situation imposerait l'utilisation d'un protocole interopérable, TLS reste un bon choix, à condition d'éviter les problèmes associés. La première recommandation est de s'assurer de la possibilité d'utiliser la version la plus récente du protocole, que ça soit côté client ou côté serveur. Cela élimine bon nombre de vulnérabilités au niveau protocole mais implique également un suivi des publications sur le sujet afin de s'assurer que seul les versions encore "sûres" sont maintenues actives. Malheureusement, dans le cas de communications avec des entités extérieures ne maintenant pas nécessairement leur système à jour, une version antérieure et vulnérable peut être nécessaire. Dans ce cas, soit les communications utiliseront une ancienne version, donc vulnérable, soit les communications avec cette entité seront impossibles. En plus de cette recommandation initiale, la configuration du système doit être prise en compte. En particulier, s'assurer que les certificats utilisés pour garantir l'authenticité de TLS ne sont pas altérés. Pour cela, une approche simple est de ne pas dépendre du magasin de certificats fourni par le système d'exploitation, mais d'embarquer un magasin de certificats propre à l'application. Enfin, l'implémentation utilisée doit aussi être maintenue à jour, afin d'éviter les vulnérabilités purement logicielles qui pourraient, alors que la version du protocole utilisée est correcte, entraîner une baisse de la sécurité des communications ou des systèmes communicants impliqués. En prenant en compte ces recommandations, on peut éviter les risques majeurs liés à l'utilisation de TLS. D'autres inconvénients persistent, notamment la possible divulgation d'information d'identité du client, ou le besoin important en bande passante.

1.2.3 Communications internes

Le modèle des communications internes permet d'avoir un meilleur contrôle et une meilleure connaissance des deux extrémités d'une communication sécurisée. La levée de la contrainte d'interopérabilité permet de s'affranchir d'un bon nombre de vérifications et donc de réduire la phase de négociation au minimum nécessaire pour l'établissement d'une connexion sécurisée. Cette réduction a deux avantages : elle nécessite moins de bande passante et présente moins d'éléments variables réduisant ainsi la surface d'attaque à cette étape. En effet, contrairement au cas des communications externes, les deux extrémités de la communication sont maîtrisées par la même entité, ce qui permet de faire des suppositions fortes sur les protocoles et algorithmes supportés de chaque côté. De plus, en considérant que tous les éléments constituent un même ensemble applicatif, il est raisonnable de penser qu'un système de mise à jour cohérent existe et permet d'appliquer les correctifs à toutes les instances de l'application. Cet aspect élimine les risques de conflits de versions, où une instance pourrait tenter d'utiliser une version plus ancienne du protocole. Les deux seuls cas à détecter lors d'une négociation sont donc extrêmement simplifiés : soit l'entité à l'autre extrémité de l'échange utilise le même protocole, soit ce n'est pas le cas.

L'idée de développer un protocole dont la principale caractéristique "visible" serait l'absence d'interopérabilité va à l'encontre des solutions courantes consistant à favoriser l'interopérabilité. Cependant, malgré l'absence de solutions existantes pour ce problème, le besoin est réel : les applications ayant un besoin de communications internes, sans interopérabilités, sont une réalité. Le cas typique de ces communications internes est une application constituée d'un serveur et d'applications clientes fonctionnant sur un protocole spécifique ; il n'y a pas de besoin de compatibilité avec des clients génériques, car les deux extrémités forment un tout. Pour ces applications, l'utilisation de solutions génériques comme TLS impose un surcoût en ressources, tout en apportant des risques de sécurité liés à cette généricité. C'est pour cela que le protocole SVC [26] (section suivante, 1.3) a été développé. Sa conception a été guidée autant par des contraintes d'exploitation que par des contraintes théoriques.

1.3 Le protocole SVC

Le protocole SVC[58] a été conçu explicitement pour répondre au cas des communications internes. En effet, l'absence de solutions existantes permettant d'appliquer l'ensemble des propriétés de sécurité souhaitées nécessite l'apport d'une

26. Secure Virtual Connector, connecteur virtuel sécurisé, protocole de communication sécurisée dédiées aux communications internes des applications

solution adaptée. Il faut un protocole de communication ayant un coût minimum en termes de bande passante et de temps de calcul, fournissant toutes les propriétés de sécurité décrites en introduction et pouvant aussi bien s'intégrer à des applications existantes que servir de base à la construction de nouvelles applications.

Pour cela, nous avons conçu le protocole SVC à partir de zéro avec les objectifs suivants : restreindre autant que possible les fonctionnalités disponibles, réduire le surcoût en bande passante imposé par les algorithmes de sécurité, limiter le nombre d'échanges nécessaires à l'établissement d'une connexion sécurisée et fournir l'ensemble de propriétés de sécurité décrites précédemment.

1.3.1 Motivations pour un nouveau protocole

En plus des propriétés de sécurité classiques que sont la confidentialité et l'authenticité, nous avons considéré deux propriétés supplémentaires qui améliorent la sécurité des communications : la confidentialité persistante et la protection de l'identité du client. En rendant la confidentialité persistante obligatoire, les échanges sont non seulement protégés lors de l'échange, mais aussi en cas de fuite d'informations ultérieures. En effet, si la clé cryptographique associée à un utilisateur est divulguée après qu'un échange sécurisé ait eu lieu, cela ne met pas en péril la confidentialité de cet échange. La protection de l'identité du client permet de fournir un niveau minimum d'anonymat en chiffrant les éléments qui permettraient d'identifier un utilisateur lors de l'établissement d'une connexion sécurisée. Ce chiffrement présente le même niveau de sécurité que le reste de la communication, y compris la confidentialité persistante, ce qui fait qu'il n'est pas possible a posteriori, même en s'appropriant les informations de l'utilisateur, de le relier à une communication ayant eu lieu dans le passé.

Le tableau 1.4 résume les propriétés fournies par les trois protocoles TLS, SSH et SVC. Il apparaît que TLS ne fournit pas l'ensemble des propriétés de sécurité requises et que même si SSH fournit l'ensemble de ces propriétés de sécurité, il impose une consommation importante de bande passante pour sa phase de négociation et ne peut pas être intégré aisément dans certains types d'applications.

Puisque notre objectif était de concevoir un protocole en accord avec les contraintes des communications d'applications modernes, nous avons dû prendre en compte des contraintes d'exploitation en plus de contraintes théoriques et de sécurité. Une des contraintes les plus importantes est la mobilité des utilisateurs et de leurs appareils.

En considérant les restrictions et les comportements courants observables sur des réseaux de natures différente, nous avons pu ajuster le protocole SVC pour qu'il puisse s'adapter à différents réseaux et appareils sans compromettre les propriétés de sécurité fournies. La prise en compte des contraintes des différents réseaux observés, notamment les réseaux mobiles, ont motivé la réduction à la fois du coût

Propriétés	TLS	SSH	SVC
Confidentialité	+	+	+
Conf. persistante	O	+	+
Authentification serveur	+	+	+
Authentification client	O	+	+
Protection identité client	–	+	+
Économie de bande passante	–	–	+
Facilité d'intégration	–	–	+

TABLE 1.4 – Propriétés des différents protocoles

'–' : absente, 'O' : optionelle, '+' : obligatoire

en bande passante et du nombre d'échanges nécessaires pour la négociation. Ces limitations permettent un fonctionnement sur tous type de réseau, qu'il s'agisse de réseaux classiques, à faible débit, ou avec des temps de réponses élevés.

Afin d'illustrer l'importance de ces restrictions, qu'il s'agisse des contraintes de capacité réseau ou des restrictions imposées par les propriétés de sécurité supplémentaires, deux cas d'usages sont présentés au paragraphe 1.4.1.2. Ces deux cas ont été volontairement choisis pour représenter des cas génériques de communications point à point classiques. En illustrant l'utilisation des différents protocoles de communications sécurisés dans une communication point à point, nous intégrons un grand nombre de cas pratiques, qu'il s'agisse de communication client/serveur, de communications machine à machine, de synchronisation d'instances dans un cloud numérique, etc. Ici, les paramètres essentiels motivant la création d'un protocole dédié sont la vitesse de négociation, l'occupation des réseaux utilisés et la sécurité effective des algorithmes mis en place. Ces paramètres dépendent davantage de l'environnement dans lequel ces communications ont lieu que leur finalité. C'est pourquoi nous ne présentons pas l'utilisation concrète dans des cas d'usages spécifiques, mais uniquement le comportement des protocoles de communications sécurisées dans des situations représentatives des communications courantes.

Une autre contrainte opérationnelle concerne la mobilité des utilisateurs qui a comme conséquence que le protocole doit s'adapter à des politiques de sécurité variées. Ce besoin d'adaptation a été pris en compte dans la conception du protocole afin de pouvoir être flexible notamment au niveau de l'authentification du client sans compromettre la protection de cette étape d'authentification. Cette mobilité augmente la variété des appareils ayant à établir des communications sécurisées et impose des choix formats et d'algorithmes utilisables.

Cas des réseaux restreints : Nous pouvons considérer le cas "classique" des communications comme s'appliquant à des réseaux sur lesquelles les limites en

bande-passante et en latence sont raisonnablement larges. Cependant, le nombre d'appareils mobiles (comme les smartphones ou les réseaux de capteurs) bénéficiant d'un accès réseau a explosé ces dernières années. Nous nous trouvons donc dans une situation ou une part importante des échanges opère non pas sur des réseaux classiques, mais sur des réseaux restreints, comme les réseaux mobiles. Ces réseaux peuvent avoir des restrictions bien plus importantes sur leur bande passante, leur latence, ou même sur la nature des contenus autorisés à y circuler. L'utilisation de ces réseaux n'est par ailleurs pas limitée aux appareils mobiles. Ils sont parfois aussi utilisés pour des systèmes fixes en utilisant un smartphone comme passerelle réseau. Puisque la notion de communication sécurisée implique, au-delà de la confidentialité, d'apporter une certaine garantie sur la délivrance des messages, un protocole sécurisé doit s'accommoder des restrictions des supports de communication considérés.

Politiques de sécurité : Les politiques de sécurité concernant les applications (mobiles ou non) peuvent recouvrir plusieurs aspects, allant de l'authentification des utilisateurs au stockage sécurisé des données sur un appareil client. La mobilité des utilisateurs impose de nouvelles approches notamment sur la fiabilité de l'authentification d'un utilisateur. Cette authentification pouvant avoir lieu sur plusieurs appareils de natures différentes, avec des modalités d'authentification différentes. Un exemple d'un tel système est l'utilisation d'une authentification à clé publique. Dans un tel système, nous conservons les clés cryptographiques de l'utilisateur sur son appareil. Dans le cas d'un système mobile, il faut mettre en place des solutions permettant de conserver un haut niveau de confiance dans le processus d'authentification tout en prenant en compte le fait qu'un utilisateur puisse utiliser de nombreux appareils différents pour s'authentifier. Le protocole SVC en lui-même n'a pas pour objectif d'apporter une solution à cette question. En revanche, il fournit un mécanisme de sécurité permettant à l'application qui l'utilise de mettre en œuvre de façon sécurisée un système d'authentification au niveau de l'application. Un espace est réservé lors de la négociation d'une communication sécurisée permettant à l'application d'utiliser un mécanisme d'authentification de son choix. Cet espace est transmis de façon sécurisée et confidentielle. De cette façon, l'intégration du protocole SVC dans une application existante permet de continuer à utiliser un éventuel mécanisme d'authentification existant. Cette approche permet de respecter les politiques de sécurité existante en fournissant un moyen sécurisé d'exécuter un protocole d'authentification quelconque, spécifié par l'application. Le protocole SVC est ici en charge uniquement de la protection de cet échange d'authentification, déléguant ainsi l'authentification à un protocole dédié. Nous disposons ainsi d'une flexibilité importante sur ce point sensible, sans risque de compromettre la confidentialité des informations utilisées pour l'authen-

tification.

Sécurité du système d'exploitation : Un autre élément qui a mené à la conception du protocole SVC est la sécurité des appareils. Différents systèmes d'exploitation peuvent fournir des outils de sécurité dont le comportement peut être modifié de façon significative. L'exemple le plus courant est la présence d'un trousseau de clés cryptographiques géré par le système. Si un attaquant parvient à altérer ce trousseau de clés, il peut totalement compromettre la sécurité des applications qui en dépendent. Il est important que la sécurité d'un protocole soit aussi indépendante que possible de la sécurité du système d'exploitation, en se séparant de tout élément de configuration ou tout composant logiciel qui y est lié et qui peut être altéré par une tierce partie. Une implémentation du protocole SVC doit suivre strictement cette règle et donc être auto-contenue, en incluant tous les algorithmes et éléments de configuration au niveau de l'application plutôt qu'au niveau du système d'exploitation.

1.3.2 Conception du protocole SVC

Le protocole SVC est conçu pour répondre à la problématique des communications internes (section 1.2.3). La plupart de ses éléments peuvent se présenter comme une version allégée de TLS et SSH. Ce protocole fournit cependant davantage de propriétés de sécurité, implémentant l'ensemble des propriétés de sécurité décrites dans la section 1 tout en ayant un impact minimal sur les coûts en ressources comparé aux autres protocoles.

Le protocole SVC fonctionne comme une couche intermédiaire entre la charge utile d'une application et le réseau effectif. Il n'impose pas de restrictions à l'utilisation sur la quantité de données. Les limitations liées aux algorithmes sous-jacents, comme les limites de volume qu'un algorithme de chiffrement peut traiter, sont gérées de façon transparente par le protocole. D'un autre côté, les limitations du réseau utilisé sont gérées dans la mesure du possible au niveau du protocole. En particulier les limitations sur les tailles de paquets, si applicable, sont prises en compte dans les opérations de chiffrement afin de profiter au mieux des ressources disponibles. Pour ces raisons, le protocole SVC fonctionne comme une encapsulation transparente de paquets, masquant les restrictions du réseau sous-jacent. Par conception, le protocole se situe au niveau application pour rester indépendant autant que possible des fonctions du système d'exploitation.

La seule contrainte du protocole SVC vis-à-vis du protocole d'authentification mis en œuvre est que ce dernier permette de réaliser une forme de signature numérique. Cette signature est utilisée pour authentifier, en plus des éléments d'identification, les paramètres de sécurités du protocole. De par sa nature générique, la méthode d'authentification ne dépend pas de l'utilisation d'une PKI

telle que c'est le cas pour le protocole TLS. Il est toutefois possible d'intégrer ce type de mécanisme d'authentification avec l'utilisation du protocole SVC. Cela permet d'intégrer l'utilisation du protocole SVC dans des applications existantes qui seraient basées sur un tel système.

L'étape de négociation, point central du protocole, apporte les garanties suivantes : l'existence d'un lien confidentiel entre les deux hôtes, l'authenticité du serveur, l'authenticité du client et la protection des informations d'identité du client. Les échanges suivants la négociation se placent donc dans un contexte sécurisé où les données sont correctement chiffrées et protégées contre toutes altérations. Les éléments de sécurité devant avoir une durée de vie restreinte (en particulier les clés de chiffrement) sont changées régulièrement à partir des paramètres de sécurité échangées pendant la négociation.

1.3.3 Établissement d'une connexion sécurisée

1.3.3.1 Négociation

Le protocole opère en deux phases : négociation et échange de messages sécurisés. La phase de négociation est illustrée dans la figure 1.2. Cette étape confirme en premier lieu la version du protocole utilisé par les deux hôtes. En cas de conflit de version, le protocole s'arrête. Ce cas ne se présente qu'en cas de défaut de mise à jour d'une des extrémités de l'échange, situation gérée au niveau applicatif. Le reste du premier envoi permet de définir les paramètres de sécurité à utiliser (les clés de chiffrements, les vecteurs d'initialisation,...). Ces paramètres de sécurité sont obtenus via un protocole d'échange de clés classique. Enfin, ce premier envoi permet aussi d'authentifier le message comme provenant d'un serveur autorisé. Le client ne répond que si tous les éléments sont validés. Il termine le protocole d'échange de clés, après quoi toutes les communications seront chiffrées et donc rendues confidentielles. Cela inclut les informations d'identité et d'authentification du client afin d'éviter la divulgation de ces informations. Le choix d'émettre le premier message par le serveur a été fait pour réduire la durée de la négociation sur certains réseaux à forte latence.

L'information d'identité du serveur et du client peuvent prendre une forme arbitraire, tel que requis par l'application en charge de l'authentification. Des exemples typiques d'informations d'identité sont des clés publiques (en forme brut ou sous forme de certificats), ou des identifiants de clé pré-enregistrées. Cette partie est spécifique à l'application ; le protocole SVC se contente de fournir un moyen de transfert sécurisé entre le client et le serveur.

La version du protocole sert à identifier à la fois le protocole et les algorithmes utilisés pour le chiffrement et l'intégrité. La première version du protocole SVC s'appuie sur un échange de clés basé sur le problème de Diffie-Hellman. Cet échange

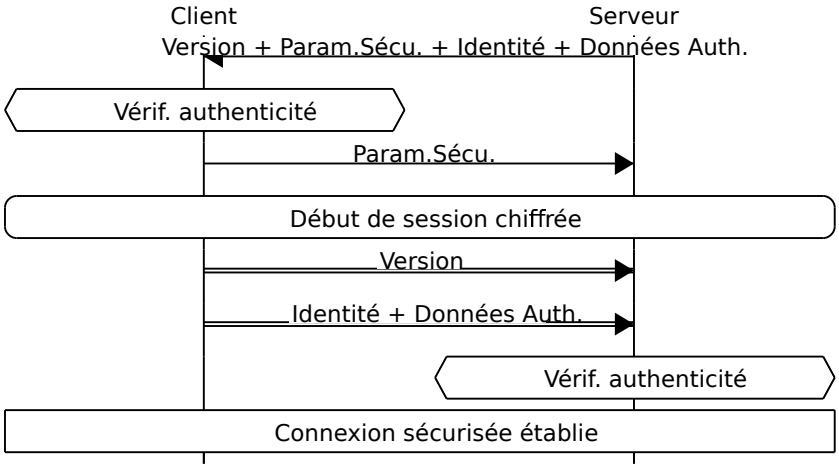

FIGURE 1.2 – Phase de négociation du protocole SVC

double flèche : contenu chiffré ; flèche simple : contenu en clair

de clés fait partie des données authentifiées lors de la négociation. Lorsque l'échange de clés se conclut, les deux extrémités de la connexion sécurisée disposent d'un secret commun K_M. Ce secret commun est utilisé pour dériver la première clé de la session sécurisée K_1. Cette clé est utilisée pour mettre en œuvre le chiffrement. Le secret K_M est également utilisé pour générer les autres paramètres de sécurité comme des vecteurs d'initialisation ou des données d'authentification additionnelles.

1.3.3.2 Échange de messages sécurisé

Après l'étape de négociation, chaque extrémité de la communication peut envoyer des messages enrobant la charge utile, comme illustré par la figure 1.3. Du point de vue de l'application, les deux instances n'ont à se préoccuper que de la charge utile. Le protocole SVC se charge du chiffrement et de l'intégrité de l'échange. Le client et le serveur gardent tous les deux un compte du volume de données échangées. Dès que les limites imposées par la sélection d'algorithmes sont atteintes (en particulier sur le volume de donnés que l'on peut chiffrer sans changer les paramètres de sécurité), l'identifiant de la clé de session courante est incrémenté et une nouvelle clé de session K_i est calculée à partir du secret commun K_M. Dès que l'une des extrémités reçoit un message indiquant un nouvel identifiant de clé, cette nouvelle clé est utilisée pour toutes les communications ultérieures. La clé précédemment utilisée est définitivement mise au rebut.

La génération des clés de session est basé sur une fonction de hachage cryptographiquement sûre et est basé sur le secret commun K_M. La génération d'une clé de session d'identifiant i se fait en calculant $H\left(K_M||i\right)$ où H est la fonction de

Inst.A → SVC S	:	Envoi de la charge utile par l'application
SVC S → SVC R	:	Envoi de l'identifiant de clé
SVC S ⇒ SVC R	:	Envoi de la charge utile chiffrée et des informations d'intégrité
SVC R	:	Mise à jour des clés (si nécessaire)
SVC R → Inst.B	:	Transmission de la charge utile à l'application

FIGURE 1.3 – Enrobage de la charge utile d'une communication par SVC

SVC S et *SVC R* représentent les extrémités du protocole SVC. *Inst.A* et *Inst.B* représentent deux instances de l'application. Les doubles flèches indiquent un contenu chiffré.

hachage et || est l'opération de concaténation. Une méthode similaire est utilisée pour calculer les autres paramètres de sécurité à partir de K_M. L'utilisation d'une fonction de hachage cryptographiquement sûre, ainsi que d'une valeur initiale dépendant des deux hôtes permet de produire des paramètres difficiles à retrouver par un attaquant. De plus, cette approche permet de conserver un haut niveau de sécurité tout en réduisant considérablement l'impact d'un changement de clé sur la bande passante utilisée par les fonctions de sécurité du protocole. En effet, un identifiant de clé peut se limiter à un octet et toutefois servir à générer tout un ensemble de nouveaux paramètres.

Le renouvellement de la clé de session peut se produire après deux conditions : soit après l'écoulement d'une durée prédéterminée, soit après le chiffrement d'une certaine quantité de données, fixée par les algorithmes sous-jacents utilisés. Conserver les mêmes paramètres de sécurité après avoir chiffré ou déchiffré un volume de données trop important peut affaiblir la sécurité de certains algorithmes autrement sûrs[44].

1.3.4 Choix des algorithmes

Les algorithmes cryptographiques utilisés à différents niveaux doivent être choisis en fonction de leurs propriétés de sécurité et de leur résistance théorique sur le long terme. Le choix d'algorithmes est ici fixé avec la version du protocole. Nous parlons des algorithmes de chiffrement, d'échange de clés et d'intégrité, car le choix de l'algorithme d'authentification est délégué à l'application appelante. Cela contraste avec les approches classiques qui proposent une liste d'algorithmes et permettent au client (ou au serveur) de choisir dans cette liste ce qu'il supporte. Ce type de choix peut mener à des combinaisons non sûres basées sur des algorithmes sûrs par ailleurs, mais qui présentent des interactions problématiques. La restriction à un jeu validé d'algorithmes permet d'une part d'éviter ces interactions malheureuses et d'autre part de simplifier la phase de négociation.

Contrairement à TLS et SSH, en se plaçant dans le cadre des communications

internes nous pouvons ici nous permettre de fixer un choix et interrompre toute négociation qui ne le respecterait pas. En effet, en ayant la maîtrise sur les deux extrémités de la connexion, nous avons la certitude que le choix "imposé" sera accepté lors d'une négociation. De plus, l'utilisation d'un ensemble restreint d'algorithmes permet de faciliter leur intégration au niveau applicatif. Cette dépendance entre les mises à jours de l'application (contenant l'ensemble des implémentations de sécurité) et les protocoles de sécurité est une caractéristique acceptable uniquement dans le modèle des communications internes, contrairement au modèle des communications externes où client et serveur peuvent disposer de versions conflictuelles d'un protocole.

La première version du protocole SVC utilise les algorithmes suivants :

— ECDH [27][41] pour l'échange de clés
— AES-128/GCM[42] pour le chiffrement
— La fonction GHASH (composante du mode d'utilisation GCM [28]) pour l'intégrité et l'authenticité
— SHA2-256[1] pour la génération des paramètres de sécurité

Ces algorithmes ont été choisis pour plusieurs raisons. L'utilisation d'opérations sur courbes elliptiques permet d'avoir un niveau de sécurité comparable avec l'arithmétique classique tout en utilisant des éléments de taille inférieure : une courbe elliptique ayant une taille de corps de 512 bits apporte une sécurité comparable à une opération basée sur le logarithme discret utilisant un groupe de 15 360 bits tout en étant plus rapide. L'algorithme de chiffrement AES [29] est l'algorithme de chiffrement par bloc recommandé actuellement pour une taille de clé d'au moins 128 bits[4]. Le mode d'utilisation GCM est une extension du mode compteur réputé sûr permettant d'apporter des capacités d'authentification des échanges. La famille de fonctions de hachage SHA2 est elle aussi le standard actuel en matière de fonction de hachage sécurisée. Les tailles de clés et de blocs présentés ont été choisis selon les recommandations actuelles en matière de sécurité. L'ANSSI [30] conseille en effet pour un système cryptographique cohérent l'utilisation d'un chiffrement symétrique utilisant une clé de 128 bits, une taille de groupe de 256 bits pour les calculs sur courbes elliptiques et une fonction de hachage de 256 bits afin de fournir une sécurité fiable pour plusieurs années. Ces recommandations rejoignent celles émises par le NIST [31][16]. Ces choix d'algorithmes

27. Elliptic-Curve Diffie-Hellman, Diffie-Hellman basé sur courbes elliptiques

28. Galois/Counter-Mode, mode de chiffrement par blocs intégrant un mécanisme d'authentification des données

29. Advanced Encryption Standard, algorithme de chiffrement par bloc le plus utilisé de nos jours

30. Agence Nationale de la Sécurité des Systèmes d'Information, chargée de proposer les règles à appliquer pour la protection des systèmes d'information de l'État et de vérifier l'application des mesures adoptées

31. National Institute of Standards and Technology, institut Américain faisant référence no-

et de tailles de clés apportent un niveau de sécurité élevé sans impact significatif sur les performances du système.

Quand ces algorithmes (ou leur combinaison) feront l'objet de vulnérabilités effectives, il sera possible de spécifier une nouvelle version du protocole utilisant un autre jeu d'algorithmes. Pour le déploiement d'une nouvelle version du protocole SVC, il suffit alors de déployer une nouvelle version de l'application l'utilisant. Cela est possible parce que l'ensemble du protocole est contenu au niveau applicatif sans dépendance avec des composants externes ou le système d'exploitation.

Nous ne mentionnons pas dans cette spécification les algorithmes permettant de réaliser la gestion éventuelle des clés publiques et des signatures numériques. Ces éléments sont fournis par l'application et sont transférés de façon transparente par le protocole SVC entre le client et le serveur. Cela permet d'être compatible avec toute politique de sécurité déployée au niveau de l'application. La seule contrainte posée sur le système d'authentification et qu'il doit permettre de produire une signature numérique, c'est-à-dire fournir une fonction prenant en entrée les données d'identité (dont le format est lui aussi libre) et les données à authentifier et produire en sortie un élément pouvant permettre de vérifier a posteriori l'authenticité. Après que l'application ait fourni au protocole SVC une telle fonction et les informations d'identité adéquates, le protocole se charge d'encapsuler ces éléments dans sa phase de négociation. Il apporte automatiquement la confidentialité sur l'ensemble des éléments identifiant du client et vérifie de façon transparente l'authenticité de la négociation. Un effet de bord positif de cette approche est qu'il est possible d'intégrer au protocole SVC une architecture d'authentification basée sur une PKI, y compris celle utilisée par TLS. Même si cette approche n'est pas recommandée, cela peut s'avérer nécessaire en termes d'utilisabilité pour intégrer le protocole SVC dans des applications existantes.

1.4 Comportement du protocole

1.4.1 Performances

1.4.1.1 Coût théorique en ressources

Le surcoût en ressources du protocole SVC par rapport à des communications non sécurisées est minimal à la fois en termes de bande passante et de nombre d'échanges. Pour une exécution normale, un seul aller/retour est nécessaire pour compléter la négociation. Le volume nécessaire pour effectuer cet aller/retour varie selon la modalité d'authentification utilisée. Les échanges suivants ont un surcoût

tamment dans le domaine des algorithmes cryptographiques

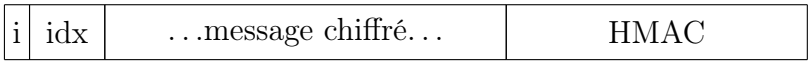

FIGURE 1.4 – Constitution d'un message du protocole SVC

en bande-passante fixe. Chaque message échangé se voit augmenté de trois éléments : un identifiant de clé (un octet), la position du paquet (trois octets), et un élément de contrôle d'intégrité (HMAC). L'algorithme GCM permet d'obtenir un élément de contrôle d'intégrité de taille variable. L'utilisation de paramètres de sécurité appropriés permet d'utiliser une taille de 12 octets pour cet élément ; il s'agit de la plus petite taille proposée pour l'utilisation du contrôle d'intégrité du GCM ne mettant pas en péril la confiance de cet élément[44]. Dans ces conditions, le surcoût en bande-passante se limite à 16 octets, en suivant le modèle de la figure 1.4.

L'étape de négociation est aussi compacte que possible. L'échange de clés est basé sur des courbes elliptiques, ce qui permet de réduire la bande passante nécessaire en conservant le même niveau de sécurité que les approches classiques. La terminaison de la négociation en un seul échange la rend efficace sur des réseaux à forte latence. Une autre conséquence de cette négociation rapide est que la renégociation sur des réseaux avec de fortes pertes de paquets ou des déconnexions régulières, est acceptable. En effet, en considérant un cas extrêmement défavorable ou seule deux échanges peuvent avoir lieu avant une perte de connexion, le deuxième échange permettra malgré tout de communiquer de façon sécurisée.

1.4.1.2 Cas d'usages

Certains éléments du protocole sont de tailles variables : tout ce qui est lié à la modalité d'authentification impacte la taille des éléments d'identité et d'authentification. Afin de pouvoir comparer les performances du protocole SVC avec les protocoles classiques que sont TLS et SSH, il faut fixer la taille de ces données. Nous développons ici deux exemples utilisant deux modalités d'authentification différentes. La première présente un cas simple de communication entre deux entités inconnues, le deuxième sert de point de comparaison avec les protocoles existants. Ces deux cas d'usages se concentrent sur l'impact des protocoles de sécurités présentés dans ce chapitre (TLS, SSH et SVC). Ils ne présentent pas immédiatement de cas d'utilisation concrets, mais deux situations permettant de couvrir de nombreux usages (communications client/serveur, communications machine à machine, etc).

Authentification a clé publique : Le premier exemple pourra servir à la mise en place d'un échange ponctuel entre deux entités proches (tels que deux smart-

phones). Pour cet exemple, nous considérons un système d'authentification à clé publique utilisant ECDSA [32] basé sur la courbe $P-521$ du NIST. Ici, ni le serveur ni le client ne se connaissent. La validation des clés peut avoir lieu de manière visuelle, avec l'affichage d'une empreinte comparable sur chacun des appareils. La taille d'une clé publique dans ce système est d'environ 150 octets et les signatures produites sont d'environ 130 octets. Afin de transmettre ces éléments de taille variable de façon contrôlée, nous utilisons un encodage simple ou leur longueur est elle-même codée sur un entier non signé de deux octets. Pour des raisons de clarté, cet exemple utilise la même courbe elliptique pour l'échange de clés. Au total, la négociation nécessite pour chaque entité : un octet pour le numéro de version, 150 octets pour l'échange de clés (ECDH), 150 octets d'information d'identité (la clé publique), et 130 octets d'authentification, plus trois fois deux octets pour la taille des champs de taille variable, soit 437 octets. La négociation complète opère donc en un aller-retour et 874 octets. Ce volume de données est adapté pour de nombreux types de réseaux, y compris des réseaux à faible capacité tel qu'un réseau de type Bluetooth, sur lequel la valeur de MTU [33] courante est de 672 octets. Sur ce genre de réseaux, chaque message de la négociation pourra être transmis en une seule fois et sera moins sujet aux problèmes de retransmission ou de congestion. Évidemment, la même chose reste vraie pour des réseaux moins limités.

Comparaison avec les protocoles existants : Afin de pouvoir comparer la négociation du protocole SVC avec les protocoles SSH et TLS, il faut définir une modalité d'authentification similaire. En ce qui concerne la négociation de SSH, effectuer la comparaison avec un système à clé publique comme décrit dans le paragraphe précédent est acceptable. En effet, SSH utilise une authentification à clé publique pour authentifier le serveur et l'utilisation d'une clé publique côté client est aussi possible. Pour permettre une comparaison avec TLS, il est plus raisonnable de considérer une authentification du serveur avec un certificat X.509. À titre d'exemple, un certificat X.509 contenant une clé RSA [34] 2048 bits fait environ 1100 octets. Il s'agit du type de certificat couramment utilisé de nos jours. Il est intéressant de noter que même si les certificats X.509 utilisés par TLS peuvent supporter une authentification basée sur courbes elliptiques, leur utilisation n'est pas encore répandue. L'authentification du client par le protocole TLS pouvant utiliser des moyens très variés et nécessitant le support d'outils spécifiques complexes, la comparaison est ici faite sans cette étape. Dans la mesure où le protocole TLS présente un coût en ressources plus important malgré l'absence d'authentification client, le

32. Elliptic-Curve Digital Signature Algorithm, signatures numériques basées sur courbes elliptiques

33. Maximum Transmission Unit, taille maximum d'un paquet transmis sur un réseau donné

34. Rivest Shamir Adleman, algorithme de chiffrement a clé publique parmis les plus utilisés au monde

Protocole	Messages échangés	Taille totale (o)
Auth. par clé publique		
SSH	16	6383
SVC	2	880
Auth. par certificats X.509		
TLS	6	5244
SVC	2	1670

TABLE 1.5 – Comparaisons entre SVC, SSH et TLS

Valeurs expérimentales obtenues après plusieurs mesures des étapes de négociations.

résultat de cette comparaison reste pertinent. Pour permettre de comparer avec le protocole SVC, il faut toutefois noter que même si l'authentification est basée sur RSA 2048, le protocole d'échange de clés utilisé par TLS dans le tableau 1.5 est lui aussi basé sur ECDH. Pour résumer, nous avons deux scénarios de comparaisons :

- Comparaison avec SSH : authentification client et serveur via leurs clés publiques respectives basées sur courbe elliptique
- Comparaison avec TLS : authentification serveur uniquement via un certificat X.509

Le tableau 1.5 résume les différences au niveau de la négociation entre les trois protocoles. Il y a un net avantage pour le protocole SVC, que ce soit en nombre de messages échangés ou en bande passante nécessaire. La conséquence est que dans le cas de communications utilisant un support ayant une qualité de service faible (telle qu'un réseau mobile), les processus de négociations nécessitant plus d'échanges et plus de bandes passantes sont susceptibles d'échouer avant d'avoir pu se terminer.

1.4.2 Analyse de sécurité

L'utilisation d'algorithmes connus comme fiables et de bonnes pratiques en ce qui concerne leur combinaison permet d'assurer que les opérations de chiffrement sont sûres. De plus, la mise en place du chiffrement avant la transmission d'informations sensibles par le protocole permet de protéger l'identité des utilisateurs. L'ensemble du système s'appuie sur la fiabilité du processus d'authentification, fourni par l'application appelante. Cette section traite des attaques courantes sur les protocoles de communication sécurisé rendues impossibles pour le protocole SVC de par sa construction.

1.4.2.1 Fuite d'information par retour d'erreur

Pour le protocole SVC, lorsqu'une erreur se produit, qu'il s'agisse d'un problème d'intégrité, d'identifiant invalide ou autre, il n'y a pas de notification de la cause exacte de l'erreur. L'absence de détails à ce niveau est intentionnel pour éviter toute fuite d'information. En effet, indiquer une erreur comme l'impossibilité de valider l'intégrité d'un paquet peut permettre de fabriquer de faux messages qui seraient alors considérés comme "valides"[44]. C'est pour cela que le protocole ne propose pas de mécanisme de récupération en cas d'erreur. Pour reprendre la communication, il faut établir une nouvelle connexion sécurisée. Cette approche présente l'avantage que chaque connexion utilise de nouveaux paramètres de sécurité et réduit fortement le risque de manipulation malveillante. Dans la mesure où l'étape de négociation du protocole est très rapide, même sur des réseaux de faible capacité, cette solution est acceptable.

1.4.2.2 Man in the Middle (homme au milieu)

Une attaque de type MitM est une attaque dans laquelle un attaquant s'insère entre deux entités souhaitant communiquer et peut observer et modifier le contenu des échanges. Le cœur du protocole SVC est l'algorithme d'échange de clés Diffie-Hellman[55]. Cet algorithme est vulnérable aux attaques MitM. En effet, dans la version classique, il n'y a pas d'élément d'authentification. Si l'algorithme Diffie-Hellman permet d'échanger une clé entre deux entités α et β sans en garantir l'authenticité, un attaquant γ dans une position de MitM peut alors tromper α et β en établissant deux échanges de clé : un entre α et γ, un autre entre γ et β. De cette façon il pourra intercepter de façon transparente toutes les communications entre α et β.

Dans le cas du protocole SVC, l'échange de clés est lui-même authentifié. En supposant un mécanisme d'authentification sûre (l'utilisation de cryptographie à clé publiques des cas d'usages précédents), l'algorithme Diffie-Hellman ne peut aboutir qu'avec une information d'authentification valide, ce qu'un attaquant ne peut pas reproduire. Le pire scénario dans le cas d'un MitM est celui d'un DoS dans lequel l'attaquant déclencherait un grand nombre de négociations. Ce genre d'attaque peut être détecté au niveau applicatif, car l'étape de négociation doit pouvoir se terminer en un aller-retour. Le problème se contourne en limitant le nombre de tentative de négociation qu'un client donné peut réaliser sur une courte période.

1.4.2.3 Rejeu

Il est parfois possible pour un attaquant d'enregistrer un échange à un moment donné et de construire une nouvelle "négociation" basée sur cet enregistrement,

afin de prendre la place du serveur ou du client. Lorsque ce genre d'attaque est utilisé pour prendre la place d'un client, l'attaquant peut usurper l'identité d'un utilisateur, identité obtenue lors d'une précédente session. Dans le cas de l'usurpation du serveur, cela permet de tromper les clients potentiels en leur faisant croire qu'ils communiquent avec une entité valide et peut mener à la fuite d'informations sensibles.

Ce type d'attaque n'est pas possible lors de l'utilisation du protocole SVC, l'échange de clés est authentifié durant la première étape de la négociation et ne se termine que si le client peut authentifier avec certitude le serveur. Cette authentification, combiné avec l'utilisation de l'échange de clés Diffie-Hellman, fait que même si un attaquant pouvait se placer dans la position du serveur et rejouer une négociation précédente, les informations sensibles (l'identité du client et les échanges ultérieures) restent indéchiffrables. Un attaquant ne peut donc pas usurper la position du serveur ou du client. Le risque le plus élevé ici est l'envoi par le client d'informations d'identité à un usurpateur. Cet envoi serait déjà chiffré et ne présente donc pas un risque majeur.

1.4.2.4 Récupération de clé de sessions

Les clés de session sont des clés cryptographiques utilisées pour le chiffrement effectif des données lors d'une communication sécurisée. Il s'agit de clés éphémères qui sont obtenues au terme de la phase de négociation. Elles ne sont liées à aucun élément permanent conservé par le client ou le serveur au-delà de leur utilisation au cours d'une communication.

Cependant, l'utilisation d'une même clé durant de longues périodes augmente le risque d'une récupération par un attaquant, en particulier si ce dernier dispose d'informations sur la nature du contenu échangé. Au minimum, une attaque par force brute sur une communication enregistrée "hors-ligne" est toujours possible. Quelle que soit la méthode employée, si un attaquant parvient à récupérer une clé de session, l'ensemble des données (passées et futures) chiffrées avec cette clé sont compromises.

Pour réduire l'impact de ce risque, les clés de session utilisées par le protocole SVC sont renouvelées régulièrement en fonction de deux critères : quand le volume de données traités dépasse un seuil acceptable pour une seule clé, ou lorsqu'un délai trop long s'est écoulé. Le changement de clé lors du dépassement d'un volume de données réduit l'efficacité d'une attaque sur ce jeu de données. En effet, la récupération d'une clé est une opération extrêmement coûteuse. Une expérimentation récente[13] présente l'utilisation de matériel hautement parallèle pour réaliser une attaque en force brute sur différents algorithmes de chiffrement, dont AES en effectuant 560 000 tentatives par secondes, soit une part négligeable des $2^{1}28$ clés possibles pour AES-128. Bien qu'encore non applicable de façon efficace, les

performances de ces systèmes et leur nombre ne va faire que croître, rendant réaliste la compromission de ce chiffrement par des attaquants disposant de moyens conséquents. C'est pour cela que réduire l'efficacité de cette attaque est un enjeu important ; le temps de calcul nécessaire à la récupération d'une clé est multiplié par autant de clé différentes utilisées tout au long de la communication. Pour des raisons similaires, les clés ne sont pas conservées au-delà d'un temps donné, afin de protéger les communications ultérieures.

Dans le cas du protocole SVC, le changement de clé ne nécessite pas de communication préalable entre les deux entités. Seul le changement de l'identifiant de clé, situé en début de message, est nécessaire. Cette approche permet de changer de clé aussi souvent que nécessaire en minimisant l'impact du protocole sur l'utilisation réseau.

1.5 Perspectives et recommandations

Une même application peut nécessiter les deux types de communication décrits dans la section 1.2. En effet, il n'est pas rare de déployer dans un même système une partie interopérable, communiquant avec des applications extérieures et une partie interne, permettant des échanges et des synchronisations entre différentes instances d'une application. C'est sur ce modèle que fonctionnent de nombreux services distribués. L'objet du protocole SVC n'est pas de remplacer les protocoles existants mais d'attirer l'attention sur le besoin mal satisfait des communications internes en y apportant une réponse adaptée. Une application ayant un besoin fort d'interopérabilité aura toujours la nécessité de s'appuyer sur des protocoles dédiés tels que TLS. C'est pourquoi il est important d'attirer l'attention sur les faiblesses de tels protocoles et d'apporter une solution adaptée pour les autres cas.

La version actuelle du protocole SVC a été conçue pour répondre à des problématiques strictes et réalistes pour le modèle des communications internes. Il reste certains points qui peuvent donner lieu à des améliorations sensibles du protocole, tout en restant dans l'espace de contraintes imposées. Parmi ces points, nous pouvons citer un double système d'authentification qui permettrait d'authentifier à la fois l'utilisateur et son terminal de façon efficace, ainsi que l'ajout de mécanismes de compressions pour optimiser l'utilisation de la bande passante.

Le double système d'authentification permettrait d'alléger la tâche des protocoles d'utilisation en les intégrant davantage dans le processus de négociation d'une communication sécurisée. Cette intégration permettrait au protocole d'authentification de transmettre en toute sécurité des informations sensibles, sans avoir à établir un second canal sécurisé. Pour réaliser cela, il faudrait effectuer des changements dans le déroulement du protocole SVC. Il est probable qu'il faudrait davantage d'échanges, mais surtout il deviendrait alors nécessaire de disposer d'une

authentification du terminal “client”. Cette première authentification serait indépendante de l'utilisateur et permettrait de certifier un appareil pour une utilisation sécurisée.

L'ajout de mécanismes de compression ou de traitement de l'information permettrait de réduire efficacement le coût en bande passante sans compliquer le développement de l'application appelante. En effet, un mécanisme de compression est plus efficace sur les données non chiffrées, ce qui signifie qu'il est plus intéressant de l'appliquer avant la transmission par le protocole SVC. Cependant, effectuer cette compression avant de transmettre les données au protocole de communication sécurisée signifie que les tests d'intégrité utilisés par ce dernier s'appliquent à une donnée transformée, ce qui peut introduire des redondances avec un test d'intégrité lié à l'algorithme de compression. De plus, l'intégration de mesures visant à réduire l'utilisation de la bande passante par les communications est intéressante à réaliser dans un protocole dédié aux communications, plutôt que dans le cœur d'une application métier. Afin de répondre à ces questions d'efficacité tout en améliorant le protocole SVC, il serait donc intéressant d'étudier quels algorithmes de compression peuvent s'appliquer dans cette situation.

Pour conclure ce chapitre il est important de noter qu'une partie de la sécurité du protocole SVC s'appuie sur l'utilisation de modalités d'authentification fiable, pour authentifier le côté serveur comme le côté client. Les cas d'usages proposés sont basés sur des problèmes pratiques variés, couvrant à la fois un usage de proximité et une utilisation à grande échelle. Dans le deuxième cas il s'agit de systèmes nécessitant une mise en place préalable d'éléments spécifiques (comme la dissémination de certificats de clé publique). Il est souhaitable de disposer de moyens d'authentification plus variés, tout en conservant la contrainte de compatibilité avec des systèmes cryptographiques. Le chapitre suivant présente une solution permettant d'utiliser des modalités d'authentifications variées avec un système cryptographique.